PROYECTO FIN DE CARRERA

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE INGENIERÍA MECÁNICA

ÁREA DE INGENIERÍA DE ORGANIZACIÓN

DISEÑO DE UNA HERRAMIENTA FLEXIBLE PARA RESOLUCIÓN DE PROBLEMAS DE ASIGNACIÓN DE INFRAESTRUCTURAS MEDIANTE SIMULACIÓN

ALUMNO: JORGE EDUARDO GONZÁLEZ LÓPEZ

TUTOR: MIGUEL GUTIÉRREZ FERNÁNDEZ

OCTUBRE DE 2012

Título Proyecto Fin de Carrera: ***Herramienta informática para modelado flexible de problemas de asignación de infraestructuras***

Autor: ***Jorge Eduardo González López***

Tutor: ***Miguel Gutiérrez Fernández***

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 30 de Octubre de 2012 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

SECRETARIO VOCAL PRESIDENTE

AGRADECIMIENTOS

Después del largo y trabajoso camino que ha supuesto la culminación de este proyecto, no puede quedar sin mencionar todas aquellas personas que lo han hecho posible y que han contribuido de forma esencial.

En un proceso tan largo y con tantos momentos difíciles, el apoyo de algunas personas ha sido imprescindible para poder continuar día a día. Ellos saben bien que la carga de este proyecto no ha caído sólo sobre mí, y que en gran medida han tenido que soportar esos momentos dificíles en los que todo se pone cuesta arriba. A ellos les agradezco profundamente, y siempre les estaré agradecido por su apoyo incondicional a la vez que necesario.

Sin ellos nunca habría llegado al final de este camino. Para que algo funcione bien, todas sus partes tienen que hacerlo, y es que el apoyo moral y sentimental ha sido una pieza fundamental para avanzar cada segundo. Muchas veces lo más importante son los pequeños detalles, y en esto me han dado la fuerza suficiente para ir poco a poco sin desesperar hasta este momento. Gracias, porque han tenido que aguantar todas las frustraciones, enfados, y desánimos, sin recibir nada a cambio.

Ellos son parte de este proyecto, y ellos han realizado conmigo cada una de las tareas. Ellos son mis padres, que sufren si yo sufro y que ríen si yo río. Siempre han hecho lo imposible por mi, y agradezco infinitamente cada esfuerzo que han hecho. Sin ellos y su esfuerzo hoy no estaría escribiendo esto. Mi hemano, qué decir de él, toda la vida juntos, superando las cosas juntos y compartiendo buenos y malos momentos. Él me entiende siempre y me apoya siempre, y sin él nada sería posible; siempre juntos. Mi novia, cada día ayudándome y aguantándome, apoyándome y luchando conmigo para que todo se haga posible. Sin ella nada sería igual, supone la motivación que me ha ayudado en esos malos momentos, y que me ayuda todos los días a ver las cosas de otra manera y con un mañana mejor. Ellos son la pieza clave que realmente hace funcionar la máquina, sino nada funcionaría ni siquiera mal.

Me gustaría acabar este proyecto con una palabra para todos ellos, GRACIAS.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN

En esta primera parte de introducción se van a tratar cuatro puntos. En primer lugar, se detalla el origen del proyecto, es decir, los antecedentes sucedidos en la línea de investigación del proyecto, así como el marco en el que se sitúa el mismo y algunas características propias del proyecto. En segundo lugar, se describirán los principales objetivos que se persiguen con la realización del mismo y el modo de abordar dichos objetivos. El tercer punto consiste en la presentación de la estructura que se va a seguir a lo largo del documento para exponer todos los aspectos importantes desarrollados.

Por último, se incluirá un estudio económico y la planificación por tareas de forma genérica que se ha llevado a cabo a lo largo de todo el proyecto.

1.1 ANTECEDENTES

Dada la naturaleza del proyecto y el papel determinante de la situación inicial en su desarrollo, es de especial importancia ahondar y detallar sus antecedentes. Así, con el fin de facilitar su comprensión, en este primer apartado se comienza describiendo el marco en el que se plantea junto a los trabajos anteriores que se toman como punto de partida y sientan las bases sobre las que se ha desarrollado.

La realización del proyecto surge en el marco de una línea de investigación mantenida durante los últimos años en el Área de Ingeniería de Organización de la Universidad Carlos III de Madrid. Esta línea de investigación se plasma en particular en el proyecto de investigación del Programa Nacional llevado a cabo en el trienio 2006-2008, *Sistema avanzado de ayuda a la toma de decisiones para la gestión hotelera (DPI2005-09132-C04-04),* y su continuación natural desde 2009 hasta 2012 a través del proyecto *Optimización de la asignación de infraestructuras de servicios mediante simulación - sectores hotelero y sanitario (DPI2008-04872).*

Este último proyecto es en el cual se enmarca el presente proyecto fin de carrera, así como el proyecto fin de carrera de Fernando José Carrasco, *Herramienta informática para modelado flexible de problemas de asignación de infraestructuras*, 2011, que cabe considerar como proyecto base del presente proyecto y destacar por tanto como antecedente directo. De hecho, este trabajo se plantea como una continuación natural, una revisión y ampliación del proyecto base, cuyo objetivo se enuncia en la memoria como:

Desarrollar una herramienta informática fácilmente expandible que permita modelar de forma flexible, rápida e intuitiva problemas de asignación de infraestructuras a clientes, pertenecientes a un segmento o agrupación, que acceden por distintos canales para realizar una reserva.

La línea de investigación marco tiene un foco en la aplicación de técnicas Revenue Management (RM) al problema de asignación de infraestructuras apuntado en el objetivo anterior, y plantea su resolución con la asistencia de un sistema de soporte a la decisión (Decision Support System, DSS). El problema así enunciado se toma también como temática objeto para el diseño planteado en este proyecto. Asimismo, se adoptan de inicio tanto la estrategia para tratar este tipo de problemas como las técnicas RM que se consideran para su tratamiento.

El concepto de *Revenue Management* (RM), también conocido como *Yield Management* o Gestión de Ingresos, tiene un amplio significado, pero en su definición clásica se expresa como "Vender el producto correcto, al cliente correcto, en el

momento correcto" [3]. El concepto comenzó a aplicarse a comienzos de los años 70 en el sector de las aerolíneas comerciales. En este sector, puesto que está caracterizado por unos elevados costes fijos, el RM se centraba en conseguir una maximización de los ingresos. Para ello, su principal objetivo de acuerdo a Zaki (2000) era "*vender el asiento correcto, al cliente correcto y en el momento correcto, al precio correcto para maximizar el beneficio*".

Podemos decir, desde una amplia visión, que el RM es el proceso de reaccionar y anticiparse al comportamiento de los consumidores, de manera que se obtenga el máximo beneficio de cada uno. Se trata de conseguir lo máximo que el cliente esté dispuesto a dar por los servicios recibidos. Para lograr este objetivo, hay que conocer en profundidad a los clientes y sus necesidades, es decir, los diferentes tipos de clientes (segmentación), las características de cada grupo, y más concretamente qué, cuándo, y cómo requieren unos determinados recursos o servicios, así como a qué precio están dispuestos a pagar para conseguirlo.

Sin embargo, en diversos casos y en particular en el problema de asignación de infraestructuras objeto, como barrera para la implantación del RM cabe mencionar la dificultad de adaptar y particularizar los algoritmos de RM a cada proceso de negocio. Esto es debido a que cada proceso de negocio requiere obtener un algoritmo que recoja todos los aspectos específicos del negocio, puesto que pueden convertirse en el punto diferenciador o valor añadido de la organización.

El trabajo introducido en los proyectos previos ya mencionados, y el proyecto base que continúa este proyecto, trata de superar esta barrera. Para ello se pretende aprovechar la ventaja que ofrece la flexibilidad de un enfoque de modelado genérico para diseñar el modelo base de un sistemas de soporte a la decisión (DSS Decision support system) basado en RM, y dirigido a la eficiente asignación de servicios de infraestructuras (Asientos de avión, habitaciones de hotel, médicos en hospitales, etc).

En definitiva, el propósito que se persigue con esta línea de trabajo es el de diseñar un sistema de soporte a la decisión (DSS) fundamentado en la aplicación de las técnicas RM. Este proyecto fin de carrera supone un paso importante para conseguir dicho propósito y para ello aborda los objetivos que se exponen en el siguiente apartado.

1.2 OBJETIVO DEL PROYECTO

En este apartado se concretan los principales objetivos que se pretenden conseguir con la realización del proyecto, además de las líneas de trabajo que se han llevado a cabo para su consecución y los ámbitos de aplicación a los que está orientado.

Como se ha explicado en el apartado anterior, el objetivo de este proyecto viene determinado de inicio y está acotado en su ámbito, por la línea de investigación en la que se enmarca y las aportaciones de los proyectos que lo anteceden. El planteamiento de partida genérico es el de avanzar hacia el propósito último de obtener una herramienta que constituya un DSS (Decision Support System), o sistema de apoyo a la toma de decisiones, en línea con la aplicación de la metodología de Revenue Management (RM) y enfocada a la optimización de los procesos de asignación de infraestructuras a los que está destinada.

Partiendo de los resultados del proyecto fin de carrera base [2], los objetivos planteados en este proyecto permiten avanzar de forma sustancial hacia dicho propósito último. En concreto, se establece un objetivo principal y dos objetivos complementarios que validan la propuesta del objetivo principal.

OP. *Elaborar un diseño conceptual robusto y completar su implementación en una base de datos, que sirva de núcleo para el desarrollo de una herramienta informática que permita tanto modelar, como simular/optimizar de forma flexible, rápida, e intuitiva problemas de asignación de infraestructuras a clientes que acceden por un canal para solicitar determinada infraestructura.*

La validación del diseño se realiza en dos niveles, de acuerdo con los objetivos complementarios:

OC1. *Corregir, modificar y ampliar la herramienta informática resultado del proyecto base para que, de forma coherente con el diseño conceptual completo, permita, de forma flexible, rápida e intuitiva, el modelado tanto estático como dinámico de los problemas de asignación de infraestructuras objeto.*

OC2. *Validar el diseño de la capa de ejecución mediante la elaboración manual de simulaciones representativas, incluyendo el detalle de los registros de la base de datos*.

Como hemos explicado anteriormente, el proyecto está basado en la aplicación del RM como herramienta para optimizar la eficiencia de los procesos de negocio. Aunque en su origen el RM surgió como una corriente para maximizar los ingresos, nuestro proyecto está enfocado a la optimización del proceso de negocio siendo el objetivo el deseado por el usuario, y por tanto definido por éste a través de las funciones objetivo. Esta aplicación del RM, no sólo enfocada a la maximización de ingresos, es el enfoque actual y futuro que se está tratando de implantar.

La aplicación de la herramienta será más eficiente en problemas que además cumplan las siguientes propiedades, y que serán mejor detalladas en el capítulo siguiente (Apartado 2.1.1):

- Posibilidad de segmentar el mercado
- Inventario perecedero
- Demanda variable en función del momento en el tiempo
- Demanda predecible
- Demanda sensible al precio
- Capacidad relativamente fija
- Costes marginales de venta bajos y costes marginales de producción altos

El objetivo es tanto modelar el subconjunto de problemas objetivo, como de optimizar los parámetros deseados por el usuario a través de simulación, que es el método elegido para llevar a cabo la optimización de los procesos de negocio definidos.

Para una eficiente optimización, y como veremos más adelante, el objetivo es que el usuario sea capaz de definir de forma flexible tres elementos fundamentales:

1. Debe recoger las características específicas del proceso de negocio, como el tipo de clientes que hay, las infraestructuras que puede solicitar cada tipo de cliente, las etapas por las que pasa un cliente desde que surge la necesidad hasta que finaliza su

paso por el proceso de negocio, y muchas más características propias del entorno del negocio y que serán recogidas por la herramienta informática a lo largo del proceso de definición del problema.

2. La variable/es que se van a optimizar mediante el proceso de simulación. Dichas variables son los parámetros del proceso de negocio a optimizar de acuerdo a unos objetivos (siguiente elemento). Las variables pueden ser desde número de médicos óptimo, hasta el precio de un tipo de habitación de hotel, o el instante de tiempo en el que se deben cambiar los precios para cumplir con los objetivos (Función objetivo).
3. Las variables se han de optimizar de acuerdo a la Función Objetivo. La función objetivo es evaluada tras cada simulación, y recoge tanto restricciones que debe cumplir el sistema como la evaluación del objetivo a optimizar. Como explicaremos más adelante, la herramienta incorpora la posibilidad de definir una función objetivo múltiple, que consta de varias funciones objetivo a optimizar de acuerdo a una prioridad dada.
   La función objetivo puede ser la maximización del beneficio, la minimización de costes, la minimización de pacientes muertos en un hospital, o que los ingresos sean siempre mayores que los gastos (función objetivo de comparación), entre otras.

Por tanto, la herramienta recogerá estos tres aspectos fundamentales para la optimización del problema.

Para crear un DSS eficiente, éste debe componerse de tres partes fundamentales, que a su vez son los módulos que componen la estructura del proyecto y son:

I. Una base de modelos.
II. Una base de datos.
III. Una aplicación software encargada de la gestión y el tratamiento de la base de datos conforme a los modelos que la sustentan, y que supone la interfaz de usuario para la definición de los problemas de asignación de infraestructuras.

Otro de los objetivos importantes del proyecto es la implantación de un ***sistema de tratamiento de plantillas***. Gracias a este sistema, que involucra tanto la base de datos como a la aplicación software, el usuario podrá guardar y cargar plantillas de modelos previamente definidos. De esta manera, el usuario podrá de forma rápida crear un nuevo modelo basado en una ya existente. Este objetivo ha sido marcado para otorgar más rapidez y flexibilidad a la herramienta

Los tres módulos del proyecto, y para ajustarse al objetivo buscado, deben cumplir una serie de propiedades o requisitos:

- El modelo conceptual de base debe ser robusto y con cierta flexibilidad para admitir futuras expansiones y mejoras que se puedan plantear. El modelo conceptual tiene en cuenta las características generales fundamentales del subconjunto de problemas a los que está destinado, con objeto de recoger en la medida de lo posible el diseño específico de dichos problemas (Pre-personalización, será explicado en el apartado 2.1.3). Respecto al modelo conceptual, el proyecto se inició con un modelo de partida (Gutiérrez y Durán, 2011), que se explicará en el segundo capítulo, y que durante el desarrollo de la investigación fue necesario ir modificando.

- Puesto que se trata de un proyecto de investigación, en todo momento debe estar orientado a la introducción de posibles mejoras, modificaciones o expansión. Por

este motivo, es necesario que la arquitectura de la herramienta este basada en tres capas: Modelo conceptual, base de datos, e interfaz con usuario y gestión de base datos. Esto permite la gran ventaja de en caso de alteración de una de las capas, optar por el aprovechamiento de la mayor parte de las dos capas restante, facilitando enormemente el proceso de desarrollo y expansión futura. Además, esta propiedad tiene una segunda ventaja, pudiendo utilizar los mismos modelos y base de datos en caso de optar por implantar la aplicación en otra plataforma.
Gracias a esta propiedad, se puede actuar sobre una de las tres capas y realizar modificaciones con independencia de las restantes. En todo proceso de investigación y desarrollo, donde al avanzar en el proceso surgen continuas necesidades de cambio en las partes previas, esta organización modular parece necesaria para garantizar un desarrollo más eficiente.

- Respecto a la capa de datos, y teniendo en cuenta que supone la representación de los modelos y que sirve como base para la aplicación software, con el fin de garantizar una fiel reproducción de los modelos y con ello demostrar que los modelos son válidos, es necesaria llevar a cabo una metodología para la traducción de los modelos a la base de datos. Esta metodología ha sido implantada, y a lo largo del proyecto de podrá ver la equivalencia entre los distintos modelos y la parte de la base de datos correspondiente.

- Con respecto a la interfaz gráfica con el usuario (aplicación software), debe ser abordada con el objetivo de ser flexible, rápida, e intuitiva, proporcionando al usuario la posibilidad de ajustar el mayor número de parámetros posibles para que se adapten al problema que se quiere definir y optimizar. Esto hace referencia de nuevo a la propiedad que se tratará en el siguiente capítulo, ***personalización***, y que trataremos de maximizar, siempre teniendo en cuenta que la herramienta está diseñada para un subconjunto de problemas y por ello debe incorporar un grado de restricción a la hora de definir parámetros. Por lo tanto, se pretende que el usuario pueda avanzar por el proceso de definición y simulación del problema de forma rápida, intuitiva y lo más flexible posible de acuerdo a los modelos establecidos.

En resumen, el objetivo del proyecto es desarrollar una herramienta informática que permita de forma flexible, rápida, e intuitiva tanto definir como optimizar por simulación un subconjunto de problemas de asignación de infraestructuras a clientes, pertenecientes a un segmente o agrupación, que acceden por distintos canales para realizar una reserva. La herramienta tiene que recoger las características específicas del proceso de negocio a optimizar, las variables a optimizar, y la función objetivo que se pretende optimizar. Además la herramienta está constituida por tres capas, modelos, datos, e interfaz, que deben cumplir los requisitos citados arriba.

1.3 ESTRUCTURA DEL DOCUMENTO

Tras una introducción de los antecedentes y los objetivos del proyecto, este apartado especifica la estructura que sigue el presente documento con el fin de definir los distintos módulos en los que se basa para su fácil seguimiento. La finalidad de este apartado es servir como guía, ofreciendo una visión de la estructura lógica que sigue el proyecto y la relación entre los diferentes apartados.

El documento está formado por tres bloques principales para un total 8 capítulos. Estos tres bloques están dispuestos de manera lógica para facilitar la comprensión y estructura del proyecto. Puesto que se trata de un proyecto de investigación que ha requerido numerosas modificaciones, la elaboración del proyecto no ha sido un proceso lineal, aunque de forma genérica ha seguido los pasos que marca la estructura del documento.

El **primer bloque** está compuesto por los capítulos 1 y 2, y presenta tanto el marco de perspectiva general en el que se encuadra el proyecto como sus objetivos y el marco teórico de partida.

- El capítulo 1 consta de una introducción del marco general del proyecto, así como los antecedentes sucedidos en la misma línea de investigación y los objetivos principales del proyecto. Incluye también un análisis económico del proyecto y una planificación de tareas generales abordadas durante el proyecto y su dedicación.
- El capítulo 2 incluye el marco teórico, es decir, se describen los fundamentos teóricos en los que se basan los modelos conceptuales de la herramienta, y se incluye una descripción de los modelos desarrollados en los dos proyectos antecesores y de los que el actual proyecto es continuación. Este capítulo también contiene cómo se ha llevado a cabo el diseño de la herramienta y la descripción de las herramientas informáticas empleadas para tratar cada una de las partes en que se compone el proyecto (Modelos teóricos, base de datos, y aplicación software).

El **segundo bloque,** y que constituye el núcleo del proyecto, se compone de los capítulos 3, 4, y 5. Este bloque trata en profundidad las tres partes básicas del proyecto, como son los modelos conceptuales, los modelos de base de datos y su implementación, y el diseño e implementación de la herramienta software. Estos tres capítulos están directamente relacionados, ya que los modelos conceptuales son la base de las otras dos partes, y a su vez la base de datos es un elemento fundamental para la aplicación. Esta relación entre los tres capítulos hace que los apartados en los que cada uno está dividido sean semejantes y relacionados.

- El capítulo 3 expone detalladamente todos los *modelos conceptuales* desarrollados durante el actual proyecto, ya que casi la totalidad de los modelos de partida han sido modificados o cambiados por completo. Este capítulo está dividido en varios apartados, ya que, como veremos más adelante, los modelos conceptuales de base están divididos en varias capas o niveles jerárquicos, cada uno de los cuales dotado con sus características y modelos. Así pues, el capítulo incorpora los apartados: *Metamodelo de proceso de negocio, Modelo de proceso de negocio e Instancia de proceso de negocio, Modelo de Atributos, Modelo de estados, Modelo de Función Objetivo, Modelo de conjunto de ejecuciones, y modelo de ejecución.*
- El capítulo 4 incluye todos los *modelos de la base de datos*, que suponen la traducción de los modelos conceptuales a una base de datos. Por tratarse de una traducción de los modelos, de acuerdo a una metodología, el capítulo está dividido en apartados similares al capítulo anterior.
- El capítulo 5 incorpora el diseño e implantación de la *aplicación*. En este apartado se explican todos los formularios que sirven de interfaz con el usuario, así como la utilidad de cada uno de ellos. Los apartados en los que se divide este capítulo están en línea con los dos capítulos anteriores, ya que la relación entre los capítulos 3, 4, y 5 es directa.

El **tercer bloque** se compone de los capítulos 6, 7, y 8. En este bloque, el capítulo principal es el 6, pues incluye un ejemplo de uso de la herramienta, tanto en la fase de definición del problema a través de la aplicación, como la fase de simulación del proceso. En este caso, el ejemplo está basado en el proceso de negocio de un hospital.

- El capítulo 6, como hemos comentado, presenta un ejemplo de uso de la herramienta aplicado a un modelo de negocio de un hospital. En el ejemplo se incluye los distintos pasos a seguir para la definición del problema, tanto características del negocio, como variables a optimizar y función objetivo, y finalmente un ejemplo del proceso de simulación en base a los parámetros definidos. Por lo tanto, este capítulo presenta de forma ilustrativa un ejemplo práctico y real de la utilidad del proyecto.
- El capítulo 7 trata de las conclusiones substraídas tras la realización del proyecto, así como las posibles mejoras y expansiones futuras que se pueden aplicar al proyecto y que se han identificado.
- Finalmente, el capítulo 8 es la bibliografía, donde se referencian los libros y fuentes de información consultados y utilizados para la realización del proyecto. La elaboración de la bibliografía ha sido realizada siguiendo las pautas que establece la norma ISO 690-1987, admitida internacionalmente para la creación de referencias bibliográficas.

1.4 PLANIFICACIÓN Y PRESUPUESTO

En este apartado, todavía dentro del capítulo de introducción del proyecto, se pretende hacer un análisis o valoración económica de lo que podría costar el proyecto en términos económicos. Para ello, ligado a este análisis se incorpora una planificación por tareas generales que se han realizado durante el desarrollo del proyecto, así como la duración estimada de cada una de ellas. Esta planificación es necesaria para poder hacer la valoración económica.

A modo de resumen, el proyecto ha tenido una duración aproximada de 13 meses, aunque la carga de trabajo no se ha repartido de forma uniforme entre cada mes, siendo mayor las horas de dedicación durante los 6 primeros meses y los 2 últimos. Al tratarse de un proyecto de investigación y desarrollo innovador, no se puede establecer una planificación exacta, pues constantemente ha sido necesario cambiar de una tarea a otra por la relación existente entre todas las partes. Al estar todas las partes interrelacionadas, las necesidades surgidas en una parte del proyecto han implicado cambios en otras y viceversa, lo que ha llevado a la realización de tareas simultáneamente en muchas ocasiones.

No obstante, a pesar de que es difícil presentar una planificación detallada y que se ajuste perfectamente a lo acontecido, en la figura 1-1 se presenta un diagrama de Gantt que muestra las tareas generales y subtareas que se han realizado durante el proyecto. Se puede considerar que es una buena estimación y con un nivel de detalle óptimo. Además, la lista detallada de tareas y subtareas se muestran en la tabla 1-1.

Tareas desarrolladas durante el proyecto
Fase de estudio inicial
Planteamiento del marco del proyecto y alcance
Estudio del Marco Teórico y Antecedentes
Estudio de Modelos previos con detalle
Estudio Proyecto anterior con detalle
Fase aprendizaje de herramientas
Aprendizage de bases de datos con Access
Aprendizaje de modelado con UML
Aprendizaje de modelado en IDEF1X
Aprendizaje de programación con Visual Basic
Desarrollo del Proyecto
Correcciones en la aplicación existente
Estudio de las correcciones de código necesarias
Implementación de los cambios respecto a modelo actual
Desarrollo y cambio del Módelo de Estados
Rediseño conceptual y del Modelo de Datos
Rediseño e implementación de la Aplicación
Desarrollo y cambio del Modelo de la Función Objetivo
Rediseño conceptual y del Modelo de Datos
Rediseño e implementación de la Aplicación
Desarrollo y cambio del Modelo de Proceso de Negocio
Rediseño conceptual y del Modelo de Datos
Rediseño e implementación de la Aplicación
Desarrollo y cambio del Modelo de Instancia del Proceso
Rediseño conceptual y del Modelo de Datos
Rediseño e implementación de la Aplicación
Diseño e implantación de Plantillas en la aplicación
Diseño de tratamiento de plantillas en aplicación y en Modelo de datos
Implantación de tratamiento de plantillas
Desarrollo y cambio del Modelo de Atributos
Rediseño conceptual y del Modelo de Datos
Rediseño e implementación de la Aplicación
Desarrollo del Modelo de Ejecución
Diseño conceptual y del Modelo de Datos
Diseño de la ejecución / Simulación en Excel
Ajustes de Modelos y correcciones de la aplicación
Prueba de uso de la herramienta
Elaboración del Documento del Proyecto

Tabla 1-1. Lista de tareas generales del proyecto

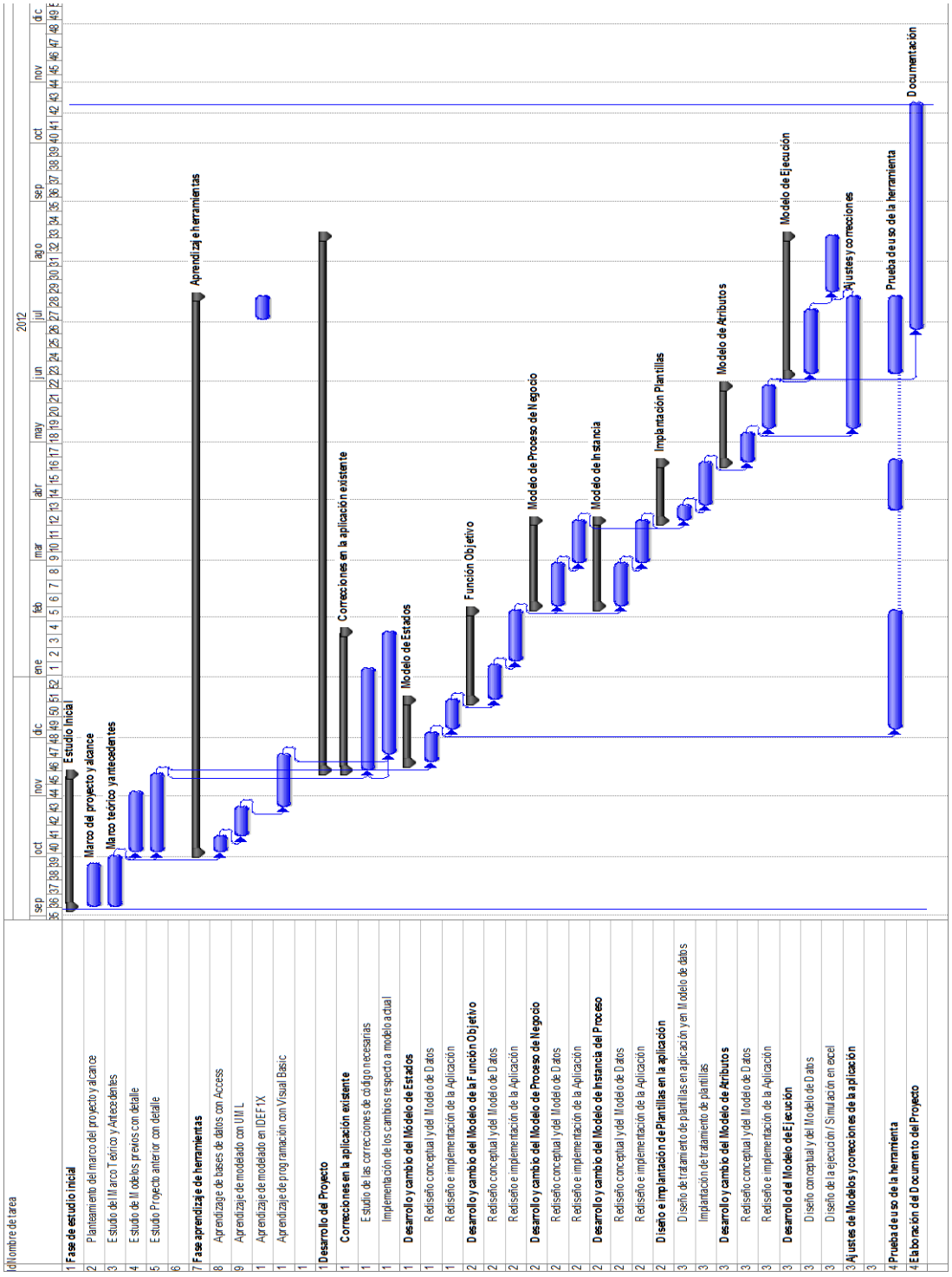


Figura 1-01. Diagrama de Gantt del proyecto

Para poder estimar el coste que podría suponer la elaboración de proyecto actual, en la tabla 1-2 se muestra la distribución de la carga de trabajo por tareas, tanto en horas de dedicación a cada tarea como en porcentaje. Esta tabla permite apreciar, de forma aproximada, en qué partes del proyecto se ha invertido más trabajo. Es importante aclarar una vez más, que al haber interrelación entre muchas de las tareas, la carga de trabajo atribuida a cada una de ellas puede tener un porcentaje que también podría ser imputado a la tarea relacionada. La figura1-2 presenta el gráfico de los datos de la tabla.

Carga de trabajo por tareas principales [Horas/hombre]			
	Tarea principal	**Horas estimadas**	**% Horas**
1	**Estudio inicial**	**200**	13,6%
2	**Aprendizaje Herramientas**	**180**	12,2%
	Desarrollo del Proyecto	**787**	53,4%
3	Correcciones en aplicación existente	150	10,2%
4	Desarrollo y cambio del Modelo de Estados	80	5,4%
5	Desarrollo y cambio del Modelo de la Función Objetivo	136	9,2%
6	Desarrollo y cambio del Modelo de Proceso de Negocio	48	3,3%
7	Desarrollo y cambio del Modelo de Instancia del Proceso	48	3,3%
8	Diseño e implantación de Plantillas en la aplicación	110	7,5%
9	Desarrollo y cambio del Modelo de Atributos	55	3,7%
10	Desarrollo del Modelo de Ejecución	160	10,9%
11	**Ajustes de Modelos y correcciones de la aplicación**	**60**	4,1%
12	**Prueba de uso de la herramienta**	**80**	5,4%
13	**Elaboración del Documento del Proyecto**	**166**	11,3%
	Total	**1473**	100,0%

Tabla 1-2. Distribución de la carga de trabajo por tareas

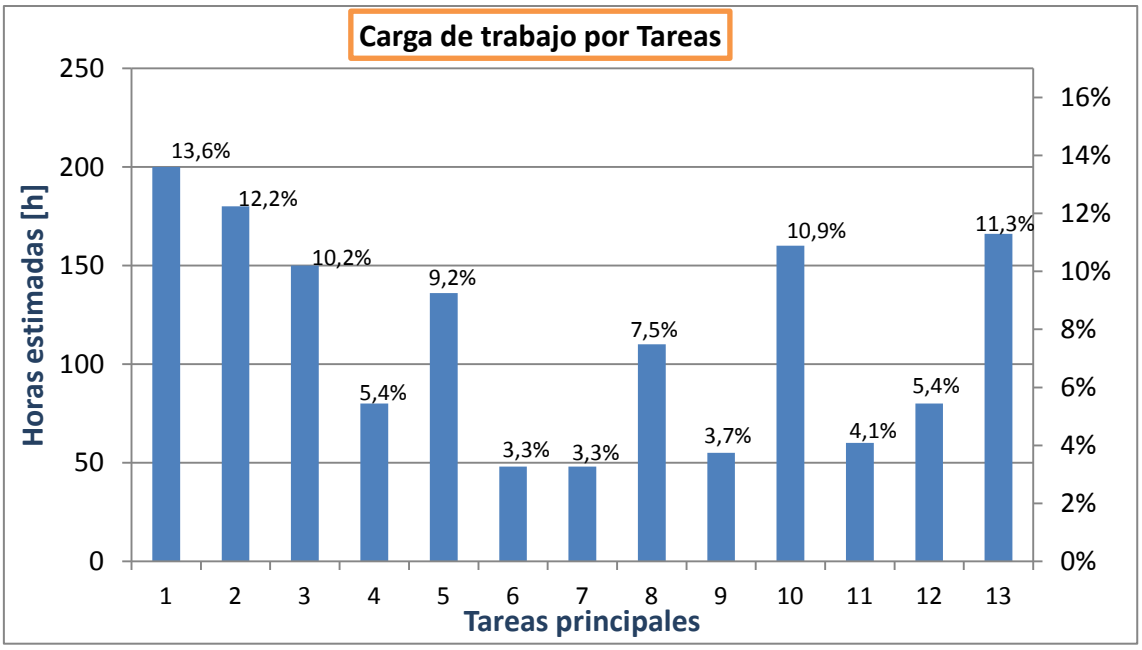


Figura 1-02. Gráfico de carga de trabajo por tareas

Una vez mostrados los datos correspondientes a la carga de trabajo requerida por cada tarea, estamos en disposición de hacer una estimación del coste que supondría la realización del presente proyecto si fuese encargado como labor de consultoría empresarial.

Al considerar que sería un paquete de trabajo, las horas correspondientes al aprendizaje de las herramientas necesarias no podrían ser cargadas como gasto al cliente y que suponen un 12,2% (180 h) de las horas de acuerdo a la tabla 1-2 anterior. El resto

de tareas sí serían imputables, incluidas las tareas de corrección de errores, ya que es un trabajo necesario para la correcta realización del proyecto.

Respecto a los costes de mano de obra, tenemos por lo tanto 1293 h de dedicación al proyecto. Si consideramos un coste horario de mano de obra para un ingeniero junior de 15 €/h, esto supondría un coste total de **20.203 €.** No obstante, este no sería el único coste de mano de obra, ya que habría que considerar las horas de soporte a la mano de obra principal, que en nuestro proyecto serían las horas correspondientes al soporte, asesoramiento y desarrollo invertidas por el tutor. Estas horas son imputables son necesarias para lograr el objetivo del proyecto, y que en caso de ser un paquete de trabajo sería una labor realizada por un segundo ingeniero, en este caso senior por ser necesario una mayor experiencia y conocimientos.

Como aproximación, podemos considerar que serían necesarias 80 h de soporte a lo largo de todo el proyecto. Considerando un coste horario de un ingeniero senior de 35€/h aproximadamente, supondría un total de 2.800 €. Esto completaría el coste total de mano de obra, y que asciende a un total de **23.003 €.**

Como medios materiales para la realización del proyecto, cabría mencionar las herramientas informáticas (software) cuya adquisición es necesaria, y un ordenador. Sin embargo, el coste de adquisición de las herramientas informáticas (Licencias) se considera como imputado en los costes generales. Sin embargo, el coste correspondiente a las horas de aprendizaje de las herramientas no se considera como imputable al proyecto. Por su parte, si consideramos un coste del ordenador de 1000 €, con un periodo de amortización de 4 años y un valor residual nulo, esto supone un coste equivalente a la amortización de 1 año, que es el periodo de duración del proyecto. En conclusión, tendríamos un coste material cercano a **250 €**.

Por último, como adición a los costes de mano de obra y costes materiales, habría que añadir una cantidad para cubrir los costes generales derivados de la actividad de realización del proyecto. En este bloque de costes entrarían conceptos como costes de desplazamiento al cliente, reprografía, luz, y otros costes que pudieran surgir. Podemos estimar que esta cantidad asciende a un 15% del coste directo de mano de obra, es decir, a los 23.003 € de mano de obra hay que sumarle un 15% de gastos generales. Por lo tanto, se puede aproximar el coste de gastos generales a **3.450 €.**

Sumando los costes de mano de obra, costes materiales, y costes generales de acuerdo a las asunciones realizadas, resultaría un presupuesto de **26.703 €** (Tabla 1-3).

CONCEPTO	COSTE
Mano de obra Directa	**23.003 €**
Mano de obra Junior	20.203 €
Mano de obra Senior	2.800 €
Coste Material (Ordenador)	**250 €**
Gastos Generales	**3.450 €**
	26.703 €

Tabla 1-3. Tabla resumen de costes

CAPÍTULO 2

MARCO TEÓRICO

Este capítulo sigue perteneciendo al primero de los tres bloques en los que se divide el proyecto. Tras exposición de los antecedentes del proyecto, de sus objetivos, y de la planificación que se ha llevado a cabo, este capítulo detalla el marco teórico en el que se basa el proyecto actual, y las herramientas empleadas.

En el primer apartado se describen los fundamentos teóricos del proyecto, divido en dos partes básicas. Por un lado el concepto de Revenue Management, y por otra parte los sistemas de soporte a la toma de decisiones (DSS). Además, en este primer apartado se incluirán las implicaciones de estos conceptos para el actual proyecto. En un segundo apartado, se explica el marco teórico de partida, que consta de los principios teóricos y modelos conceptuales heredados de los proyectos previos, y que sirven de base teórica para el proyecto actual. Aunque la gran mayoría de los modelos de partida que se presentan en este capítulo han sido modificados o rehechos en el desarrollo del proyecto, los principios teóricos sí han permanecido en su mayoría como cimientos para este proyecto.

En el tercer apartado de este capítulo, se explica el diseño de la herramienta y los módulos de los que se compone. Finalmente, el cuarto apartado presenta las distintas herramientas cuyo uso ha sido fundamental y necesario en la elaboración del proyecto.

2.1 FUNDAMENTOS TEÓRICOS

En este apartado se presentan algunos fundamentos teóricos, que ya han sido introducidos en el primer capítulo, y que serán más detallados a continuación. Estos fundamentos engloban tres partes, los conceptos de Revenue Management y su aplicación, los sistemas de soporte a la toma de decisiones (Decision Support Systems), y por último algunas implicaciones que tienen estos fundamentos para el proyecto. Cada uno de estos tres puntos será descrito en un apartado diferente.

2.1.1 REVENUE MANAGEMENT

Para poder desarrollar los contenidos del presente proyecto, primero es necesario exponer ciertos conceptos desarrollados en la línea de investigación marco y el proyectos base, según se describió en el capítulo de introducción. A este respecto, además de otras referencias, se incluyen algunos contenidos del trabajo de Gutiérrez y Durán, *Model Base Design for Decision Support Systems in Revenue Management: Applications to Hotel and Health Care Sectors*, 2011.

Este apartado se centra en la descripción del concepto Revenue Management y sus posibles aplicaciones y beneficios. En primer lugar es necesario recordar que desde una amplia visión, el RM es el proceso de reaccionar y anticiparse al comportamiento de los consumidores, de manera que se obtenga el máximo beneficio de cada uno. Se trata de conseguir lo máximo que el cliente esté dispuesto a dar en cada momento por los servicios recibidos. Para llevar a cabo una buena aplicación de los sistemas de Revenue Management hay que estudiar y conocer en profundidad a los clientes y sus necesidades, es decir, los diferentes tipos de clientes (segmentación), las características de cada grupo, y más concretamente qué, cuándo, y cómo requieren unos determinados recursos o servicios, así como a qué precio están dispuestos a pagar para conseguirlo.

El RM requiere de una combinación eficaz de un sistema de pronósticos con un sistema de optimización. Para un uso eficaz de RM, es necesario integrar un conjunto de técnicas como la segmentación del mercado, el control de inventarios, el pronóstico de la demanda, la gestión de precios, y todas las técnicas posibles para incrementar los ingresos. Una buena optimización de los ingresos requiere por lo tanto de la utilización de modelos dinámicos de predicción, la asignación de recursos o servicios perecederos a los diversos segmentos de clientes, canales y categorías de precios, así como el precio asociado a cada categoría.

Desde un punto de vista práctico, la aplicación de RM permite a las compañías convertir toda la información dispar recibida de marketing en decisiones tácticas, posibilitando a éstas aprovechar las breves oportunidades que ofrece el mercado.

Como hemos comentado anteriormente, está técnica surgió en el sector de las aerolíneas comerciales. En este sector se aprovechó el desarrollo de nuevas tecnologías de la información, la desregularización del mercado, así como la creciente demanda para implantar la técnica de RM dirigida a incrementar los ingresos por medio de nuevas políticas de gestión de precios. Estas nuevas políticas de precios trataban en primer lugar de decidir qué rangos de precios eran los óptimos, y posteriormente decidir cuándo cambiar el precio de los asientos, para una ruta y avión dado, con el objetivo de conseguir los máximos ingresos posibles de los clientes. Para lograr este objetivo, se requería un fundamental trabajo previo de previsión de la demanda y de las características del mercado.

Sobre la base del éxito mostrado por la aplicación del RM en el sector de las aerolíneas, otros sectores empresariales comenzaron a implantar dicha técnica. Entre los nuevos sectores que invirtieron y apostaron por esta nueva corriente de gestión, cabe destacar el sector Hotelero, que contiene características muy parejas al sector de las aerolíneas. Más adelante, sectores como el de alquiler de coches vieron una clara oportunidad en la aplicación de las técnicas de gestión de ingresos.

Como ocurre con toda herramienta, el RM no se puede aplicar a todas las empresas por igual, sino que requiere de una serie de características para ser aplicada de forma eficaz y aprovechando su máximo potencial. Sectores como el de las líneas aéreas o el hotelero, comparten una serie de características que permiten aprovechar al máximo las ventajas del RM. De acuerdo a Chávez y Ruiz [3], aunque se empieza a estudiar la posible aplicación del RM en empresas fabricantes de bienes, se trata de una técnica de utilidad demostrada para determinadas empresas de servicios que reúnen, entre otras, las siguientes características:

- Posibilidad de segmentar el mercado: Posibilidad de identificar diversos segmentos dentro de la cartera de clientes.
- Inventario perecedero: El inventario es perecedero en el sentido de que si no es consumido durante el período de tiempo que se oferta, se pierde toda posibilidad de obtener un beneficio del mismo. Además, si no es consumido se ocasiona una pérdida, que en el caso de empresas con costes fijos altos es más significativo.
- Demanda variable en función del momento en el tiempo: Típicamente las empresas de servicios se ven sometidas a fluctuaciones en la demanda más acusadas que en las empresas fabricantes de bienes.

- Demanda predecible: Para un adecuado funcionamiento del RM, se necesita disponer de datos de la demanda, tanto históricos, como previsionales, como actuales. La demanda debe ser predecible para tener información de los clientes.
- Demanda sensible al precio: Que la demanda reaccione a los cambios de precio. Esta característica es fundamental para que la gestión de precios pueda tener lugar como parte del RM.
- Capacidad relativamente fija***:*** capacidad relativamente fija a corto plazo, sólo modificable "a largo" con una inversión de capital relativamente importante.
- Costes marginales de venta bajos y costes marginales de producción altos: Empresas caracterizadas por soportar unos costes marginales bajos una vez que se alcanza el punto muerto, es decir, una vez alcanzado el volumen de ventas que asegure rentabilidad, cualquier venta extra supone costes de venta bajos. Empresas con costes marginales de incremento de capacidad altos cuando el nivel de producción iguala la capacidad de la empre, es decir, una venta adicional cuando la empresa está al máximo de su capacidad es muy costoso. Este punto está directamente relacionado con la característica explicada en el punto anterior.
  Por ejemplo, vender un asiento adicional una vez que el avión está fletado, no incurre demasiados costes extra; sin embargo, vender un asiento extra cuando el avión está completo puede suponer incurrir en los costes de preparar el vuelo de otro avión (costes altísimos).

En resumen, RM como medida de gestión que optimice el rendimiento, se aplica de manera más eficaz en empresas con instalaciones fijas costosas e inventario perecedero, en sectores con demanda variable, sensible al precio, predecible, y con clientela que posibilite su segmentación en grupos o categorías.

Cabe destacar que en un mundo empresarial como el actual, con una tendencia en los últimos años a la globalización y la liberalización de los mercados, donde la competencia cada vez es mayor, las empresas se ven forzadas a aplicar todo tipo de técnicas de optimización de procesos, orientación al cliente, y reducción de costes para ser más competitivas a la vez que rentables. La mayoría de las nuevas técnicas desarrolladas están enfocadas a una reducción de los costes, que a su vez permitan reducir los precios y aumentar la competitividad en el mercado. No obstante, la aplicación del RM aborda este problema desde el punto de vista de la gestión de ingresos y cómo maximizarlos, lo que también va enfocado a un aumento de la rentabilidad y competitividad.

Estas dos corrientes mencionadas son totalmente complementarias; es decir, pueden ser aplicadas en una empresa al mismo tiempo, ya que buscan un objetivo común a través de la optimización de las dos variables que determinan el beneficio, como son los costes (reducción de costes) y los ingresos. De hecho, tanto en la actualidad como con vistas al futuro, en las empresas en las que sea factible la aplicación de ambas técnicas se hace necesario la coexistencia y potenciación de éstas para alcanzar una buena posición en el mercado.

La implantación eficaz de RM implica la utilización de sistemas basados en algoritmos que deben reflejar el diseño del proceso de negocio, a cuya optimización se pretende dar soporte, a través de tres caminos relacionados [6]:

1. Debe recoger las características específicas del proceso de negocio, tales como qué escala de precios pueden ser asignados a cada tipo de cliente, o si un tipo de cliente

puede acceder a través de más de un canal. También debe incluir aspectos como los procesos de fijación de precios (Puede que algunos precios sólo puedan aumentar a lo largo del proceso, u otros sólo disminuir).

2. Es necesario recoger qué variable(s) deben ser optimizadas. La decisión a tomar podría ser cuándo cambiar los precios de un valor a otro, o a qué valor cambiar el precio en un momento determinado del proceso.

3. El objetivo(s) del negocio a conseguir. En sectores como el hotelero, el objetivo podría ser la maximización de beneficios.

Cada uno de los tres puntos anteriores tiene una implicación clara en nuestro proyecto, y será explicada en el apartado 2.1.3.

2.1.2 SISTEMA DE SOPORTE A LAS DECISIONES (DSS)

En este punto surge el segundo concepto fundamental, y que necesita ser explicado pues el objetivo del proyecto gira en torno a éste. Se trata de los sistemas de soporte a la toma de decisiones o más conocidos como *Decision Suport Systems (DSS)*. Los DSS son sistemas de la información basados en ordenador que sirven como apoyo a las personas en el proceso de toma de decisiones.

Hay muchas definiciones sobre los DSS, que han llevado a discrepancias acerca del alcance o requisitos que debe cumplir un DSS; sin embargo, atenderemos a la descripción desarrollada por Mark S. Silver [8], donde establece que un DSS es un sistema de información basado en ordenador que apoya a la gente involucrada en actividades de toma de decisiones. En este sentido, "apoyo" quiere decir:

1. El sistema asiste a las personas que toman decisiones para emitir un juicio; es decir, el sistema es una ayuda para la persona o personas que toman la decisión.

2. El sistema no toma la decisión; es decir, el sistema ayuda a las personas en el ejercicio de toma de decisiones pero no reemplaza la labor de dicha persona que toma la decisión final.

Anterior a los DSS, como fruto del desarrollo de los sistemas de la información y las tecnologías, surgieron los llamados sistemas de gestión de información, más conocidos como MIS (Management Information System), en los años 1960s. Estos sistemas permitían el almacenaje de la información recogida de los procesos transaccionales, y su uso en la gestión empresarial mayoritariamente para planificar y controlar. Los sistemas MIS permitían almacenar grandes cantidades de datos, que se presentaban posteriormente a los managers en forma de reportes (resúmenes) o datos más detallados según conviniese. A partir de la década de los 1970s, los MIS confluyen con otra línea de trabajo llamada investigación operativa (*Operations Research*, OR), para conformar los DSS. La investigación operativa empleaba técnicas y modelos matemáticos para asistir a las personas que tomaban decisiones en la solución de problemas. Por lo tanto, los DSS se conformaron como la unión de los enfoques de orientación a los datos (MIS) y orientación al modelo (OR) para soportar la toma de decisiones.

Acorde a lo establecido por Sprague y Carlson [9], los DSS se conforman en base a una arquitectura básica formada por tres componentes básicos: Una base de datos, una base de modelos, y una aplicación software encargada de la gestión y el tratamiento de la base de datos conforme a los modelos que la sustentan, y de la interfaz de usuario. Aquí se presenta una clara implicación para nuestro proyecto, que se compone de éstas tres parte, y que se explicará en el siguiente apartado (Apartado 2.1.3).

Como hemos visto, para que un DSS funcione de forma eficiente para resolver problemas de RM, es necesario que el algoritmo que fundamenta el DSS contenga el diseño específico del proceso de negocio. Esta característica, por la cual un DSS se adapta al diseño específico del proceso de negocio es denominada ***Pre-personalización*** [8]. Se puede distinguir entre sistemas pre-personalizados para un único entorno de toma de decisiones, o aquellos que están pre-personalizados para una clase de entornos. Esta propiedad está ligada con las características intrínsecas del DSS y los modelos en los que se apoya, y que el usuario no puede alterar, y el grado en el que éstos se ajustan al diseño específico del proceso de negocio.

Esta propiedad es difícil de evaluar pues se considera que hay muchas maneras en las que un sistema puede ser pre-personalizado. En nuestro caso, nuestro objetivo es desarrollar un DSS flexible que pueda ser aplicado a un conjunto de procesos de negocio que cumplen unas características, recogiendo lo más posible el diseño específico de dichos procesos de negocio, ajustándose a la segunda categoría mencionada anteriormente. Trataremos por lo tanto de conseguir un adecuado nivel de pre-personalización teniendo en cuenta que los modelos y el sistema han de ser válidos para un conjunto de entornos de negocio.

Otra propiedad relacionada con la pre-personalización, y de concepto similar pero con matices diferentes es la ***personalización (Customizability)***. De acuerdo a Mark S. Silver [8], es definida formalmente como el grado en el que, y la manera en la que, un DSS permite a sus usuarios especializarlo según sea necesario para que encaje en el entorno al que apoya. Se trata de la propiedad opuesta a la restricción, en la medida en la que una manera de que los sistemas restrinjan a sus usuarios es prevenirlos de las modificaciones de las características del sistema. Esta propiedad está relacionada con la capacidad habilitada al usuario para modificar o introducir características en la aplicación para adaptarla a las condiciones de entorno del proceso de negocio.

De nuevo, esta propiedad es difícil de evaluar por los mismos motivos que la propiedad anterior, pero el objetivo es lograr un equilibrio entre la restricción y la personalización.

2.1.3 IMPLICACIONES PARA EL PROYECTO

En este apartado se tratan alguna de las implicaciones que los fundamentos teóricos ya explicados tienen en el proyecto.

Respecto al concepto de Revenue Management, cabe destacar tres implicaciones asociadas a la implantación eficaz de un sistema RM. Cada implicación está relacionada con el punto equivalente tratado en el apartado 2.1.1 sobre los tres caminos de implantar eficazmente un sistema RM:

1. Para recoger las características específicas de cada proceso de negocio, tales como qué escala de precios pueden ser asignados a cada tipo de cliente, o si un tipo de cliente puede acceder a través de más de un canal; nuestro proyecto presenta unos modelos conceptuales desarrollados, con sus respectivas interfaces de usuario a través de las cuales éste irá introduciendo todas las características del proceso de negocio a optimizar.

2. Para definir la(s) variable(s) a optimizar, se recogerán a través de un formulario y quedará grabado en la base de datos de acuerdo a los modelos que la fundamentan.

3. En cuanto al objetivo(s) del negocio a conseguir, sectores como el sanitario, que emplearemos para realizar un ejemplo en el capítulo 6, no tienen un objetivo tan definido como en otros casos, pero podría ser la minimización de personas fallecidas en un hospital o la minimización de personas en lista de espera.

Por otro lado, en línea con la arquitectura básica que compone un DSS, la herramienta que se desarrolla durante el proyecto tiene como objeto servir de un DSS en sí mismo para un subconjunto de problemas que reúnen las condiciones ya explicadas. Para ello, se ha desarrollado una nueva base de modelos teóricos, como fruto del proceso de investigación desarrollado, a partir de los cuales se ha construido una base de datos gestionada por una aplicación, la cual desempeña la función de interfaz de usuario. El usuario será guiado por la aplicación, por medio de formularios, a través del proceso de definición de los distintos niveles jerárquicos que explicaremos a lo largo del proyecto.

Es importante señalar, que la finalidad principal durante el desarrollo del proyecto es obtener y confeccionar unos modelos teóricos de base que sean suficientemente robustos y flexibles para servir de cimientos para la herramienta que constituye el DSS como aplicación de la optimización de procesos de negocio. Por ello, el proceso de investigación consiste en la elaboración y optimización de los modelos teóricos de base, cuya validez y robustez quedarán demostradas a través de la implantación de los mismos en la base de datos y aplicación informática.

Respecto a la propiedad de ***prepersonalización***, explicada en el apartado anterior, en nuestro caso es la capacidad que tiene el usuario, a través de la interfaz con la herramienta informática, de particularizar los parámetros y las opciones que la herramienta permite para reflejar y ajustar las características a las propias del negocio a optimizar.

Nuestra aplicación (DSS) también emplea parcialmente la propiedad de restricción, por la que en cierto modo sólo permite recoger cierta información y hay características intrínsecas de los modelos que no pueden ser variadas. Esto es así pues, al tratarse de modelos y una aplicación válida para un conjunto de procesos de negocio, es necesario un cierto grado de restricción a la hora de ir guiando al usuario y sólo permitir la modificación e introducción de ciertas características. Este grado de restricción hace posible que la herramienta y los modelos sean válidos para un conjunto de entornos y no únicamente para un proceso de negocio. Sin embargo, como iremos explicando durante el proyecto, se ha intentado desarrollar una aplicación flexible para dar más posibilidades al usuario.

2.2 MARCO TEÓRICO DE PARTIDA

Como hemos comentado anteriormente, este proyecto supone la continuación de la línea de investigación destinada a la creación de un DSS para la optimización de asignación de infraestructuras. Para continuar con la labor de investigación, el actual proyecto comenzó tomando como base el modelo original de definición de un proceso de negocio que pertenece al trabajo de investigación llevado a cabo por Miguel Gutiérrez y Alfonso Durán, *Generic Model Base Design for Decision Support Systems in Revenue Management: Applications to Hotel and Health Care Sectors*, 2011.

Por lo tanto, en este apartado se van a detallar los mencionados modelos teóricos de partida correspondientes a cada uno de los niveles jerárquicos que se plantean y que se explicarán a continuación. Estos modelos teóricos irán acompañados de ejemplos de aplicación a dos sectores muy diferentes como son el sector sanitario y el hotelero. Estos ejemplos pueden resultar muy aclaratorios a la hora de entender la base teórica.

Como ya se ha explicado en el apartado de antecedentes y objetivo, los modelos tratan de hacer posible una óptima utilización del DSS, el cual se pretende conseguir para problemas de RM, teniendo en cuenta la barrera que supone la unión que ha de existir entre el algoritmo y el diseño específico del proceso de negocio. Puesto que se trata de conseguir modelos que sirvan para un conjunto de problemas que reúnan las características ya explicadas en la introducción, a la par que recoger el diseño específico de los distintos procesos de negocio, el enfoque para superar esta barrera es un **enfoque jerárquico de modelado genérico** de los procesos de negocio.

Un modelo de proceso de negocio (Business Process Model) define el modo en el que un elemento particular de la infraestructura es asignado a uno de sus potenciales usos. Como ejemplo, podemos tomar el caso particular de un Hotel, donde el proceso de negocio consistiría en el diseño del proceso de reserva de las habitaciones, considerando aspectos característicos como los distintos canales por los que se puede realizar la reserva, el proceso de fijación de precios (estrategia), el proceso de segmentación de los clientes, etc.

Puesto que el enfoque del modelado genérico es jerárquico, éste se compondrá de un total de cinco niveles de abstracción, a los cuales se ha llegado tras el proceso de investigación y desarrollo. Cuatro de los cinco niveles, de mayor a menor nivel de abstracción, son los siguientes:

- Nivel de **Metamodelo de Proceso de Negocio** (Business Process Metamodel). Se trata de la capa de mayor abstracción. En este nivel se define el modelo genérico de los procesos de negocio de asignación de infraestructuras, incluyendo los elementos básicos y las relaciones entre ellos para que, a través de instanciaciones, puedan derivarse el conjunto de posibles modelos de proceso de negocio.
- Nivel de **Modelo de Proceso de Negocio** (Business Process Model). Este nivel se corresponde con las instancias directas del nivel anterior.
- Nivel de **Instancia de Proceso de Negocio** (Business Process Instance). Es la instanciación directa del nivel anterior, en el sentido de que cada instancia de proceso se originará de un proceso de negocio, es decir, un proceso de negocio puede ser padre de muchas instancias de proceso de negocio. Esta instanciación se consigue por medio de la asignación de valores específicos a un subconjunto de los

elementos genéricos. Tras la asignación de estos valores y completar una Instancia, y tras la definición de la función objetivo asociada, ya está configurado un problema de asignación de infraestructuras, en el cual el subconjunto de elementos con valores asignados constituyen el subconjunto de parámetros del modelo, y los elementos sin valor asignado supondrán las variables del problema a optimizar.

- Nivel de **Ejecución de Proceso de Negocio** (Business Process Execution). Corresponde a la instancia del nivel anterior, puesto que representará el marco para el número de ocurrencias de cada instancia del proceso de negocio. Cobra especial importancia en problemas de RM en los que haya presente parámetros estocásticos, y por lo tanto adecuados para ser solucionados por simulación.

Puesto que muchos de los conceptos expuestos son abstractos, conviene establecer un paralelismo con un ejemplo correspondiente a un modelo matemático de optimización, y que también se muestra en la figura 2-1.

- En el primer nivel, nivel de Metamodelo de negocio, contaremos con un metamodelo describiendo los componentes básicos de una ecuación matemática, tales como las variables y operadores, así como las posibles maneras de combinar estos elementos para definir los dos miembros de la ecuación matemática.
- En el nivel siguiente, Modelo de Proceso de negocio, encontraremos ecuaciones genéricas de optimización, en las cuales ya se expresa qué elementos básicos las componen y las relaciones básicas entre ellos. Un ejemplo sería el siguiente:
  $f = 3a - b^2 + 4c,\ a, b, y\ c\ pertenecen\ al\ conjunto\ de\ números\ naturales +$
- En el nivel de instancia, tendremos diferentes funciones, dependiendo del valor asignado a un subconjunto de los elementos del modelo y qué elementos serán las variables del problema. Además, en nuestro ejemplo hay que concretar qué tipo de optimización se desea (maximización, minimización, o restricción). Por ejemplo:
  $$Min\ f = 3 \cdot 8 - 11^2 + 4c, siendo\ c\ la\ variable\ a\ optimizar$$
  $$ó\ \ Max\ f = 3a - 10^2 + 4 \cdot 7, siendo\ a\ la\ variable\ a\ optimizar$$
- En el nivel de ejecución, es donde tendrá lugar la ejecución de un exhaustivo algoritmo de búsqueda que proporcionará los posibles valores de las variables del problema (todos los posibles números naturales empezando con 0), hasta que la solución óptima sea encontrada. En las diferentes ejecuciones de una instancia, todos los parámetros tomarán el mismo valor, mientras que las variables irán tomando diferentes valores.

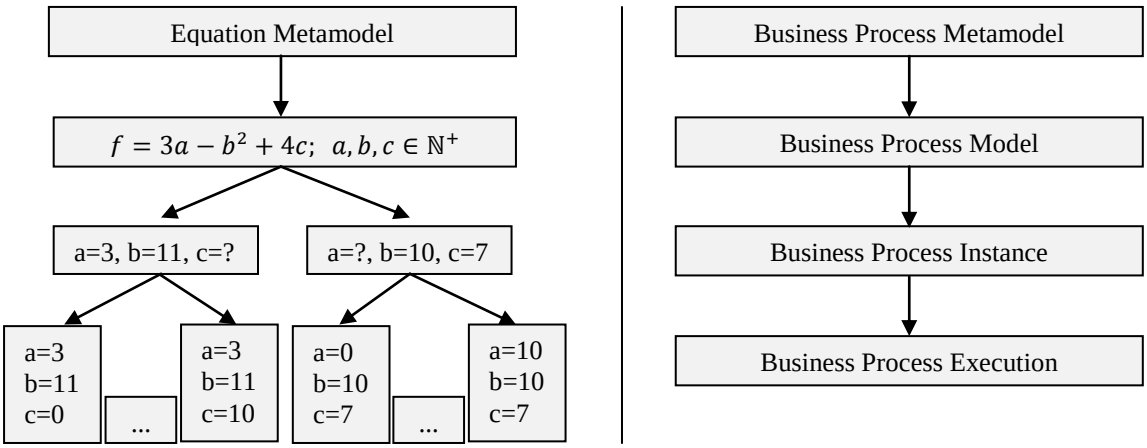


Figura 2-01. Jerarquía del modelado genérico

Estos son los cuatro niveles de partida; no obstante, durante el desarrollo del presente proyecto surgió la necesidad de añadir un quinto nivel. Este nuevo nivel se llamará Conjunto de Ejecuciones de Proceso de Negocio (Process Execution Set), y está entre los niveles de Instancia de Proceso y Ejecución de Proceso de Negocio. Este nivel se explicará en el siguiente capítulo, junto con los modelos finales.

A lo largo de todo el proyecto usaremos el lenguaje de modelado unificado UML (Unified Modeling Language) para representar los diagramas de clases correspondientes a los distintos modelos válidos para cada uno de los niveles jerárquicos del proceso de negocio. Las propiedades del lenguaje UML, así como del resto de herramientas empleadas en el proyecto, serán explicadas con detalle en el apartado 2.2.

Para profundizar más en la explicación de cada uno de los niveles jerárquicos nombrados, en los siguientes apartados ahondaremos en cada uno de ellos, mostrando el diagrama de clase asociado. Sólo presentaremos los tres primeros niveles jerárquicos, puesto que el nivel de ejecución no constaba de modelos previos.

Por último, recordar que los modelos previos han sufrido variaciones a lo largo del desarrollo del proyecto actual, motivo que hace que sólo se presenten los aspectos teóricos fundamentales, ya que en el capítulo siguiente se explicará con detalle cada uno de los modelos definitivos.

2.2.1 METAMODELO DE PROCESO DE NEGOCIO

En este apartado, y teniendo en cuenta que se trata del modelo de partida, se va a explicar con detalle el concepto de Metamodelo de proceso de negocio, siendo la primera capa de abstracción del enfoque teórico.

El Metamodelo representa el modelo de los modelos de proceso de negocio, y pretende dar soporte a todo el conjunto de problemas que quiere abordar. El metamodelo incluye los elementos básicos fundamentales, así como las relaciones existentes entre ellos, que debe contener todo proceso de negocio perteneciente al subconjunto de problemas a los que está enfocado el proyecto.

En la figura 2-2 se muestra el diagrama UML del Metamodelo y Modelo de Proceso de esta capa, y en él se pueden apreciar todos los elementos básicos, representados por cajas, y las relaciones entre ellos, representadas por líneas.

Para poder entender bien el diagrama, es necesario explicar las 7 entidades básicas del modelo, y que conforman las 7 Entidades del Modelo (Model Entities) que figuran en el diagrama.

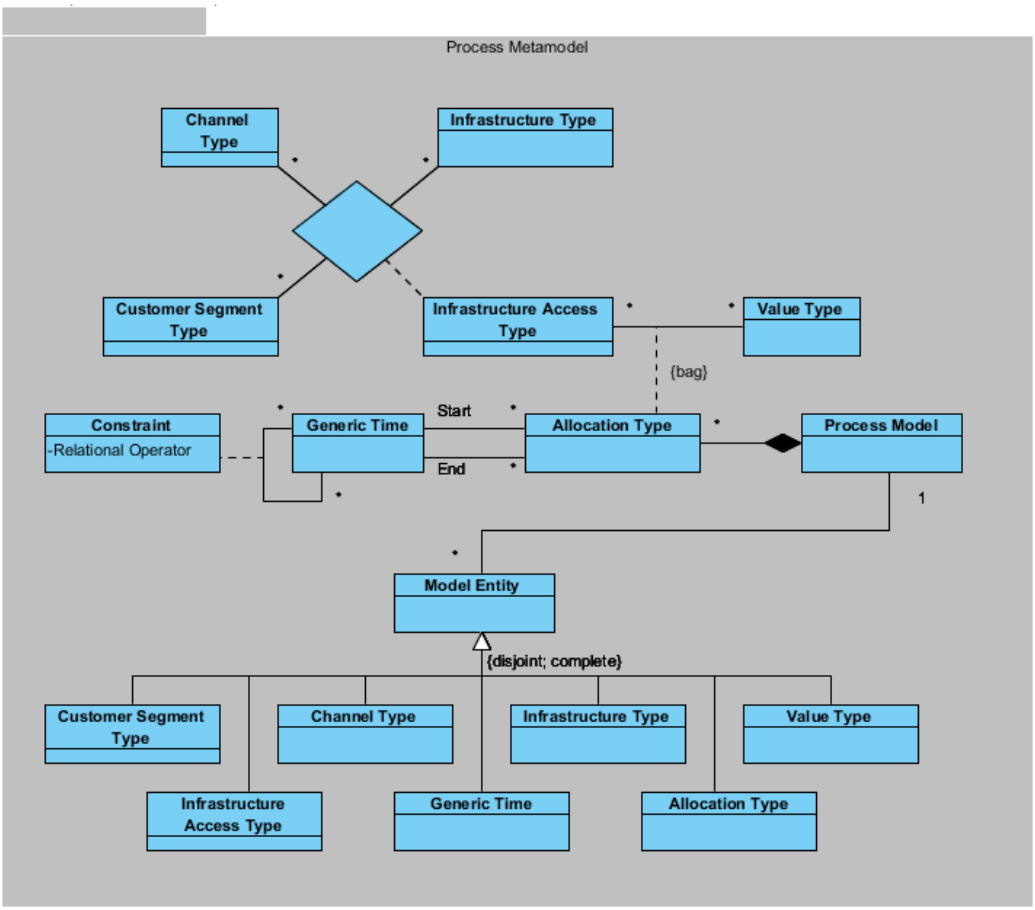


Figura 2-02. Diagrama de clase en UML del Metamodelo y Modelo de Proceso de Partida

En primer lugar, un tipo de segmento de cliente (**Customer Segment Type**) hace referencia a cada uno de los grupos de clientes que reúnen unas características comunes. Con el paso de los años, y el aumento de la competencia, la técnica conocida como segmentación del mercado se ha ido potenciando hasta convertirse en una herramienta fundamental para la estrategia de toda empresa. Mediante esta técnica, los clientes son agrupados y clasificados de acuerdos a unos parámetros que la empresa considere apropiados para distinguir distintos tipos de clientes. Estos parámetros pueden ser la edad, el nivel económico, la localización geográfica, etc. Como se explicó en el apartado de antecedentes, en RM es fundamental que los clientes puedan ser segmentados, con el fin de dirigir ofertas y acciones particulares para cada grupo de clientes y poder obtener así los máximos ingresos de cada cliente.

Otro elemento básico del Metamodelo es el tipo de canal (**Channel Type**). Se trata del medio por el cual el cliente accede al proceso de asignación de la infraestructura. El canal puede ser desde internet para reservar una habitación de hotel, hasta una ambulancia para acceder a un hospital.

El siguiente elemento es el tipo de infraestructura (**Infrastructure Type**), y que constituye la infraestructura en sí misma que es ofertada y pretende ser asignada. Generalmente en cada Proceso de negocio encontraremos más de un tipo de infraestructura, y éstas pueden ser por ejemplo habitaciones simples, o habitaciones dobles como infraestructura diferente para el caso de un hotel.

Una vez explicados estos tres elementos, podemos explicar el elemento tipo de acceso a una infraestructura (**Infrastructure Access Type**), ya que supone la combinación de la terna tipo de cliente más tipo de canal más tipo de infraestructura. Es decir, cada tipo de cliente que accede a través de un tipo de canal para que le sea asignado un tipo de infraestructura. En el caso más general, todos los tipos de cliente serían capaces de acceder a todos los tipos de infraestructura a través de cualquier tipo de canal. La aparición de restricciones en el acceso de algunos tipos de cliente, o restringiendo el acceso a cierto tipo de infraestructura sólo a través de algunos tipos de canales, nos lleva a un conjunto de problemas de asignación de infraestructuras que pueden ser abordados con el Metamodelo planteado.

Este problema de base, en el que se incluyen ciertas restricciones, se refleja en la instancia del elemento Tipo de acceso a una infraestructura. Aquí aparece un concepto fundamental, y que estará presente durante todo el proyecto, y es el concepto de **instanciación**. Instanciar una clase consiste en obtener un elemento más concreto a partir de un elemento más abstracto. Se puede encontrar una similitud con el concepto de ejemplificación, por el cual se obtiene un elemento más concreto y con características más definidas a partir de una clase más abstracta, de la cual procede. Es el proceso contrario al de abstracción. Como ejemplo aclaratorio, existe la clase coche como idea general de automóvil, y que supone un medio de transporte de mediana capacidad. Sin embargo, cada uno de los coches que tenemos en nuestras casas, como objetos diferenciales y con características más concretas, son instancias de la clase coche. Por lo tanto, la similitud con el concepto de ejemplificación se debe a que cada vez que ponemos un ejemplo estamos instintivamente instanciando.

Como habíamos introducido, cada tipo de acceso a una infraestructura se puede instanciar de forma directa en tipos de acceso a infraestructuras concretos. Como veremos a lo largo del proyecto, cada uno de los elementos básicos que componen el metamodelo serán instanciados en cada nivel jerárquico, hasta que el nivel de abstracción sea nulo, hecho que se alcanzará en el nivel de ejecución, donde todos los elementos estarán perfectamente definidos y con características totalmente concretas.

Otro elemento del modelo es el tipo de valor (**Value Type**). El tipo de valor es un elemento fundamental para la optimización de problemas de asignación de infraestructuras. En aquellos problemas cuyo fin sea exclusivamente comercial, los diferentes tipos de valor son típicamente las distintas tipos de tarifas económicas: Tarifa reducida, tarifa normal, tarifa Premium; precio de oferta, lista de precios; de temporada alta, de temporada baja, etc. Por otro lado, también existen problemas de asignación de infraestructuras donde el factor económico no es el determinante, como puede ser el sector sanitario, del cual hablaremos en repetidas ocasiones a lo largo del proyecto. En este último caso, el tipo de valor podría estar ligado al beneficio social que supone la asignación de un tipo de infraestructura a un tipo de cliente.

De acuerdo al diagrama en UML del metamodelo, cada instancia directa de tipo de acceso a una infraestructura se relaciona con un tipo de valor (Value Type), y dos tiempos genéricos (**Generic Times**), para conformar el elemento tipo de asignación (**Allocation Type**). Por lo tanto, un tipo de asignación es la combinación de un tipo de acceso a una infraestructura, vinculado a un tipo de valor, y aplicable en un espacio de tiempo determinado por un tiempo genérico de comienzo (Generic Start Time) y un tiempo genérico de fon (Generic End Time). Estos tiempos son manejados en la capa genérica del modelo de una forma abstracta, es decir, sin tomar un valor concreto.

Aunque no tomen un valor concreto, las relaciones lógicas entre los diferentes tiempos genéricos, denominadas restricciones (Constaint), han de ser definidas en esta capa mediante operadores relacionales, igual y menor o igual.

De acuerdo al enfoque de este modelo, se define un Modelo de Proceso de Negocio (Business Process Model) como un conjunto de tipos de asignación (Allocation Types). La relación que representa el rombo negro junto a Process Model significa una relación de composición. Por lo tanto, en esta capa, el metamodelo abarca, de una forma genérica, definiciones de problemas de optimización de asignación de infraestructuras.

Dependiendo de la manera en que se instancien las clases que conforman el metamodelo, se obtienen familias de modelos. Especialmente, se pueden definir familias de modelos si se encuentra alguna relación entre dos de las tres entidades: Tipo de segmento de cliente, tipo de canal, y tipo de infraestructura. También se pueden definir familias de modelos en base a qué entidad, de las tres nombradas, afecta a la definición del tipo de valor.

Respecto al primer caso, puede haber procesos de negocio en los que sólo se pueda acceder a un tipo de infraestructura a través de un determinado tipo de canal, como podría ser la reserva de asientos en un avión (Infrastructure Type) sólo a través de internet (Channel Type). También podría haber procesos de negocio en los que cierto tipo de segmento de cliente sólo pueda acceder a un tipo de infraestructura, como podría ser el caso de un enfermo leve (Customer Segment Type) que sólo pueda acceder a un médico general (Infrastructure Type).

Por otras parte, en relación al segundo caso nombrado, otra familia de modelos de optimización serían aquellos en los que el tipo de valor relacionado depende exclusivamente del tipo de infraestructura que se pretende asignar, como podría ser que en un hotel el precio (Value Type) se fije en función de si la habitación es simple o doble (Infrastructure Type). También podría ser que los clientes sean segmentados de acuerdo a la lealtad a la compañía, y en función del tipo de cliente se fija el tipo de valor, por lo que un cliente fiel a la cadena de hoteles (Customer Segment Type) reciba un mejor precio (Value Type) por reservar una habitación (Infrastructure).

2.2.2 MODELO DE PROCESO DE NEGOCIO

Como se ha explicado en definición de niveles jerárquicos, un Modelo de Proceso de Negocio (Business Process Model) corresponde a la instanciación directa del Metamodelo explicado en el apartado anterior.

Como se muestra en la figura 2-2, aparece el concepto de entidad del Modelo (Model Entity). En cada modelo de proceso de negocio (Business Process Model) existen varias entidades del modelo (Model Entities), que corresponden a los 7 elementos básicos explicados anteriormente, y que se relacionan para dar lugar al modelo. Por lo tanto, las entidades del modelo se clasifican de manera completa y disjunta en 7: Tipo de segmento de cliente (Customer Segment Type), tipo de canal (Channel Type), tipo de infraestructura (Infraestructure Type), tipo de valor (Value Type), tipo de acceso a una infraestructura (Infrastructure Access Type), tiempo genérico (Generic Time), y tipo de asignación (Allocation Type). Lo que quiere decir

que toda entidad del modelo (Model Entity) pertenece necesariamente a uno de estos tipos y nada más que a uno.

El hecho de incluir las entidades del modelo, se debe a la necesidad de vincular la existencia de cada elemento, de los 7 mencionados, a un modelo de proceso concreto. Debido a que la herramienta que se ha desarrollado está orientada al modelo de proceso, y su existencia es requisito obligatorio para definir cualquier tipo de elemento, la vinculación mostrada resulta necesaria. No tendría sentido la existencia de cualquiera de estos elementos fuera de un modelo de proceso.

Para ilustrar los conceptos expuestos, se presenta el ejemplo de un modelo simplificado del proceso de reserva de habitaciones de un hotel. Para este modelo de proceso simplificado, consideramos dos tipos de segmento de cliente (Customer Segment Type), los clientes fieles a la cadena de hoteles y el resto de clientes normales. Cada cliente que trata de hacer una reserva pertenecerá a uno de estos dos grupos. Por su parte, sólo consideramos un tipo de canal (Channel Type) a través del cual se pueda hacer la reserva, y corresponde al sitio web del hotel. Se ofrecen dos tipos de infraestructura (Infrastructure Type), habitación simple y habitación doble. Incorporamos tres tipos de valor (Value Type), Precio simple, precio doble, y precio oferta. El precio simple es el precio por día de reserva de una habitación simple. El precio doble es el precio por día de reserva de una habitación doble, y el precio oferta es el precio, tras el descuento, por día de utilización de una habitación doble. Como se aprecia en la tabla 2-1, el proceso de negocio implantado por el hotel contempla que, para clientes normales se ofertarán tanto habitaciones simples como dobles a los precios de precio simple y precio doble respectivamente, en cualquier instante de tiempo entre el tiempo genérico de inicio (T.Start) y tiempo genérico de fin (T.End). Por su parte, a los clientes fieles a la cadena de hoteles se les ofertará tanto habitaciones simples como habitaciones dobles, siendo el precio ofertado precio simple para habitaciones simples, y para habitaciones dobles puede tomar el precio doble entre el tiempo genérico de inicio (T.Start) y un tiempo cualquiera entre el tiempo genérico de inicio y el tiempo genérico de fin, que corresponde a T1, o un precio oferta entre el tiempo genérico T1 y el tiempo genérico de fin (T.End).

Customer Segment Type	Channel Type	Infrastructure Type	Value Type	Generic Start	Generic End
Normal	Web	Habitación simple	Precio simple	T.Start	T.End
Normal	Web	Habitación doble	Precio doble	T.Start	T.End
Fiel	Web	Habitación simple	Precio simple	T.Start	T.End
Fiel	Web	Habitación doble	Precio doble	T.Start	T1
Fiel	Web	Habitación doble	Precio oferta	T1	T.End

Tabla 2-1. Ejemplo de un Modelo de proceso de negocio de reserva de habitaciones de un hotel

2.2.3 INSTANCIA DE PROCESO DE NEGOCIO

La Instancia de Proceso de Negocio (Business Process Instance) es la tercera capa de abstracción del modelo jerárquico propuesto. Esta capa surge como consecuencia de la instanciación del nivel anterior mediante la definición de los atributos pertenecientes a cada uno de los elementos básicos ya explicados del modelo de proceso de negocio. Las variables de optimización son definidas en este nivel, pero no toman un valor concreto; sin embargo, los parámetros definidos reciben valores en esta capa. Esta aproximación permite establecer una taxonomía de problemas de asignación de infraestructuras, basado en la definición de las variables y el conjunto de parámetros. Para exponer con claridad el concepto, siguiendo con el ejemplo anterior de asignación de habitaciones de un hotel, es importante destacar que aún tratándose de un proceso de negocio simple puede dar lugar a diversos enfoques de optimización, es decir, a varias instancias de proceso de negocio. Dos ejemplos son los siguientes:

- Dada una distribución de llegadas para cada tipo de clientes, el número de habitaciones de cada tipo que son ofrecidas, el precio simple, doble, y de oferta, y suponiendo que no hay restricciones de acceso por el canal (web), se puede determinar el valor óptimo del tiempo T1 (variable) para que el beneficio del hotel sea máximo.
- Dada una distribución de llegadas para cada tipo de clientes, el número de habitaciones de cada tipo que son ofrecidas, el tiempo T1 al cual cambia el precio, el precio simple, y asumiendo que no hay restricciones de acceso por el canal, se puede determinar el precio doble y el precio oferta óptimos para maximizar el beneficio del hotel.

Como hemos explicado, ambos casos corresponden al mismo modelo de proceso de negocio, por lo que constituyen instancias de este proceso de negocio. En la figura 2-4 se muestra el diagrama UML perteneciente al nivel de instancia de proceso de negocio, desarrollado por Miguel Gutiérrez y Fernando José en el proyecto previo. En el diagrama se puede ver que en cada instancia de proceso de negocio (Business Process Instance) existen varias entidades de la instancia (Instance Entities), y que corresponden a las instancias de las entidades del modelo (Model Entities), o elementos básicos del modelo ya explicados en el nivel anterior (apartado 2.1.1). Como muestra el diagrama, las entidades de la instancia se clasifican de manera completa y disjunta como segmento de cliente (Customer Segment), canal (Channel), conjunto de infraestructura (Infraestructure Set), valor (Value), acceso a una infraestructura (Infrastructure Access), tiempo (Time), y asignación (Allocation). El hecho de ser una clasificación completa y disjunta significa que toda entidad pertenece necesariamente a uno de estos tipos y sólo a uno.

Cabe destacar que la relación entre la entidad de la instancia y la entidad del modelo en una generalización de la relación de instanciación, entre los elementos arriba mencionados (Instance entities) y sus padres (Model Entities). Esta relación se expresa como *Is instance of*, y su significado es el siguiente: las instancias de las clases arriba mencionadas (Instance Entities), son instancias de las instancias de las clases del nivel superior (Model Entities). Al tratarse de un concepto complejo, el siguiente ejemplo ayudará a su comprensión (Figura 2-3): en nuestro modelo, las instancias de segmento de cliente serán instancias de las instancias de la clase tipo de segmento de clientes. De esta manera, la clase cliente mayor de 25 años es una instancia de la clase tipo de segmento de cliente; por su parte, una instancia de segmento de cliente es un cliente

mayor de 25 años concreto, y a su vez este cliente es una instancia de la clase cliente mayor de 25 años por ser más concreto (menos abstracto).

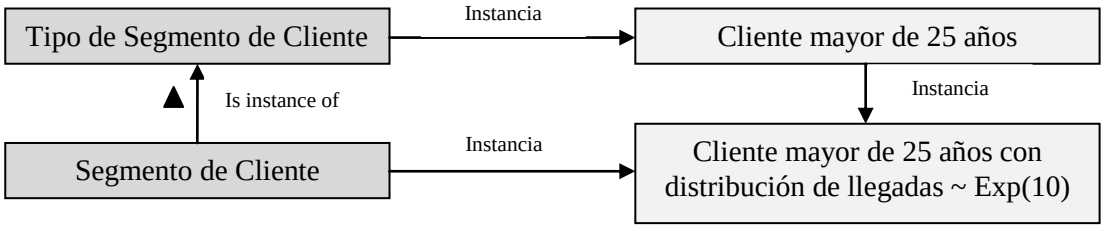


Figura 2-03. Ejemplo de significado de la relación "Is instance of"

Debido a esta relación, las restricciones que imperaban en el nivel superior siguen vigentes para cada una de las instancias directas de cada una de las entidades. En el diagrama UML se muestra como el acceso de los clientes de un segmento, a través de un canal, para reservar una infraestructura se conoce como acceso a una infraestructura (Infrastructure Access). En línea con la relación explicada en el párrafo anterior, la entidad de la instancia acceso a una infraestructura es hija del padre tipo de acceso a una infraestructura, que es lo mismo que decir *Is instance of* tipo de acceso a una infraestructura. Puesto que hereda las restricciones del nivel superior, si no existía la posibilidad de reservar un tipo de infraestructura (Infrastructure Type) a través de un tipo de canal (Channel Type), las instancias de los conjuntos de infraestructura (Infraestructure Sets) que son instancias de la instancia de ese tipo de infraestructura (infrastructure Type) no pueden reservarse a través de las instancias de los canales (Channels) que son instancias de la instancia del tipo de canal mencionado (Channel Type).

Las relaciones expresadas en el diagrama son del tipo uno a muchos, lo que quiere decir que por ejemplo en una misma instancia de proceso puede existir más de un segmento de cliente, o que un mismo tipo de segmento de cliente puede instanciarse en varios segmentos de clientes que pertenecen a instancias de procesos distintos.

Por su parte, la entidad de la instancia asignación (Allocation) es el resultado de asignar un valor (Value) a cada instancia directa de la clase acceso a una infraestructura (Infrastructure Access). Cada asignación de un valor está vigente durante un intervalo de tiempo comprendido entre un tiempo inicial o comienzo (Start) y un tiempo final (End). Estos tiempos pueden tomar un valor concreto en este nivel, o bien ser definidos como variables del problema. Además, es condición necesaria que los tiempos cumplan las relaciones de restricción definidas en el nivel superior (Constraints).

En el diagrama de la figura 2-4, se puede ver como algunas entidades de la instancia tienen asociados atributos pertenecientes a dichas clases de esta capa. Estos atributos representan algunos de los más significativos, siendo algunos necesarios en el proceso de definición de una instancia de un proceso de negocio para un correcto funcionamiento del DSS de acuerdo a los modelos desarrollados. El DSS, desarrollado en el proyecto, ayuda a la definición de atributos de entidad en este nivel. Estos atributos pueden ser introducidos por el usuario de acuerdo a las necesidades de cada problema de asignación de infraestructuras, y serán almacenados de forma automática en la base de datos correspondiente. Por lo tanto, cada instancia de un proceso de modelo de negocio constará de unos atributos necesarios para que el DSS funcione de acuerdo a los modelos base, y de los cuales hablaremos con más detalle en el capítulo siguiente, y unos atributos definidos por el usuario en base a las necesidades del mismo.

Estos parámetros pueden ser definidos como variables del problema, y serán pues optimizadas, o como parámetros.

Finalmente, el DSS asiste en la definición de la función objetivo, construida mediante combinaciones matemáticas de un subconjunto de parámetros y variables.

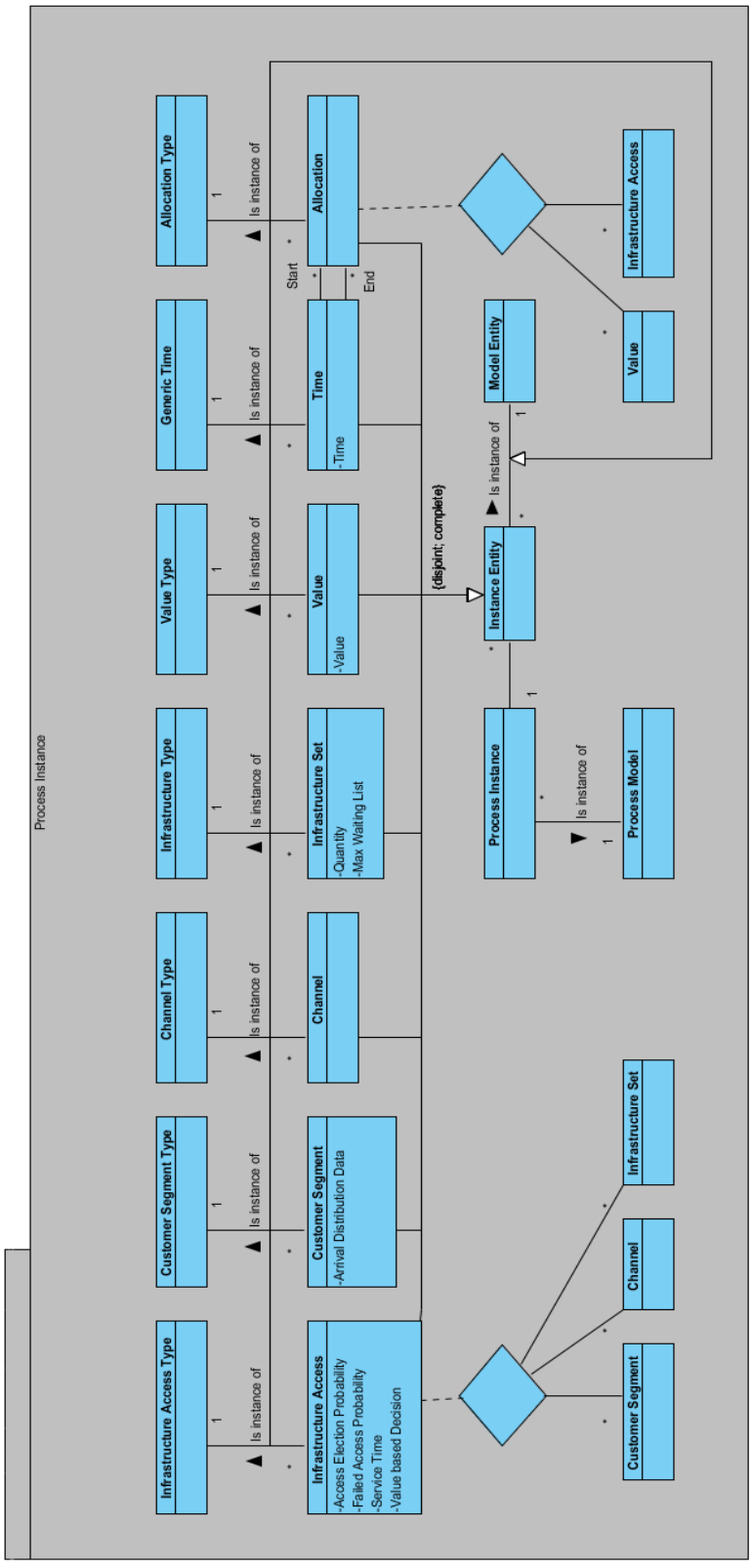


Figura 2-04. Diagrama de clase UML de Process Instance en el modelo de Partida

2.3 DISEÑO DE LA HERRAMIENTA

Tras haber explicado el marco del proyecto, los objetivos, y el enfoque basado en un modelo jerárquico en el que se basa el proyecto, este apartado explica las partes de las que se compone la herramienta desarrollada y que sirve de DSS.

En primer, hay que remarcar que la herramienta desarrollada está alineada con el objetivo del proyecto, por lo que la herramienta pretende ser un software que permita, de forma lo más flexible posible e intuitiva, definir y resolver problemas de asignación de infraestructuras con las características ya explicadas, donde un cliente, perteneciente a un segmento, accede a una infraestructura a través de un canal.

La herramienta funciona de interface con el usuario, al que asiste durante la definición del problema y su optimización. El funcionamiento de la herramienta, junto con la base de datos relacionada, supone la demostración de que los modelos teóricos de base son consistentes y robustos, ya que tanto la base de datos como la herramienta basan su arquitectura y funcionamiento en los modelos teóricos. Por este motivo, la herramienta está formada por cinco módulos, cada uno de los cuales se aborda a través de diversos formularios, y que cubren tres de los cinco niveles jerárquicos del modelo conceptual, el cual se explicará en profundidad en el siguiente capítulo.

Puesto que el actual proyecto supone la continuación de la línea de investigación, y más concretamente continúa el trabajo realizado en los dos proyectos explicados en el capítulo 1 en el apartado de antecedentes, la herramienta fue comenzada en el proyecto antecesor y desarrollado por Fernando José Carrasco. Durante el actual proyecto se han realizado todas las correcciones necesarias para un correcto funcionamiento, modificaciones para adaptarla a los nuevos modelos y requerimientos, incorporación de nuevas funcionalidades, e incorporación de formularios nuevos, para confeccionar la herramienta final. Únicamente por motivos de alcance del proyecto, no se ha podido implementar en la herramienta los módulos pertenecientes a los dos últimos niveles jerárquicos del modelo, correspondientes al nivel de **Conjunto de Ejecuciones** (Execution Set), y el nivel de **Ejecución** (Execution) o simulación. El motivo es sólo por alcance, puesto que tanto los modelos teóricos, como la base de datos, además de un ejemplo de ejecución (simulación), de estos dos niveles sí se han realizado.

- Nivel jerárquicos de metamodelo y modelo de proceso de negocio:
  - Módulo de definición de Procesos de negocio. Este módulo está basado en los modelos conceptuales de los dos primeros niveles jerárquicos del enfoque, ya explicados en el apartado 2.1. Este módulo permite la definición de procesos de negocio por medio de la definición de las entidades del modelo, es decir, los tipos de de segmentos de clientes, tipos de canal, tipos de infraestructuras, tipos de acceso a una infraestructura, tipos de valor, tiempos genéricos, restricciones temporales y tipos de asignación.
- Nivel jerárquico de instancia de proceso de negocio:
  - Módulo de definición de atributos. En este módulo, como parte de la creación de una instancia de proceso de negocio, se asiste al usuario para que defina los atributos asociados a cada entidad o elemento del modelo, es decir, los atributos asociados a los tipos de clientes, a los tipos de canal, etc. Este módulo también sirve para definir la naturaleza de cada atributo, definiendo el tipo de atributo de acuerdo a unas categorías permitidas y que veremos en el siguiente capítulo, así como definir si se trata de una variable o un parámetro, caso en el que se distingue entre parámetro estático (valor fijo durante ejecución) o dinámico.

- Módulo de definición de estados. Este módulo surge como respuesta a la definición de la dinámica de la instancia de proceso, así como a la necesidad de poder definir la/s función/es objetivo con más posibilidades y de forma más realista. Este módulo de compone de la definición de los distintos estados en los que puede encontrarse una asignación (Allocation), de los posibles cambios de estado que pueden suceder, y de la caracterización de los eventos que provocan dichos cambios de estado.
- Módulo de definición de función objetivo. Se trata del módulo en el que se definen la/s función/es objetivo del problema de asignación de infraestructuras, y en base a la/s cual/es se optimizarán las variables. Cada función objetivo está formada por una expresión matemática, si se pretende maximizar o minimizar, o de dos expresiones matemáticas relacionadas por un operador lógico si se trata de una restricción o condición. Este módulo ofrece la posibilidad de definir varias funciones objetivo para cada instancia de proceso de negocio, tanto de maximización, como de minimización o de expresión lógica, aportando esta opción gran potencia al DSS a la hora de optimizar las variables.
- Módulo de definición de atributos dinámicos. Como explicaremos en el siguiente capítulo, en el apartado de definición de atributos en la instancia de proceso, pueden existir atributos de carácter dinámico. Estos atributos pueden ser dinámicos porque pueden cambiar de valor a lo largo de cada ejecución, o porque pueden tomar un valor fijo distinto en cada una de las ejecuciones. Por lo tanto, en este módulo se caracterizan los atributos que pueden cambiar a lo largo de cada ejecución, definiendo cuándo y cómo pueden cambiar.

2.4 HERRAMIENTAS DE DESARROLLO DEL PROYECTO

En este apartado, como se ha descrito en el capítulo 1 en las tareas realizadas a lo largo del proyecto, se describen las cuatro herramientas esenciales de desarrollo que ha sido necesario aprender en profundidad y emplear para la realización del proyecto: el lenguaje de modelado de datos IDEF1X, Microsoft Access para implementación de la base de datos, el lenguaje de modelado UML, y el lenguaje y entorno de programación Visual Basic. Puesto que el aprendizaje y entendimiento de todos los matices y propiedades de las herramientas mencionadas ha requerido del estudio de un amplio número de conceptos, y su explicación bien podría ocupar un proyecto entero, en este apartado sólo se expone una introducción de los aspectos fundamentales.

2.4.1 IDEF1X

Como se ha expuesto en apartados anteriores, para que la herramienta funcione de acuerdo a los modelos teóricos desarrollados, y por lo tanto constituya la demostración de que dichos modelos son válidos, es necesario que el modelo de la base de datos, con la cual interactúa la herramienta, sea un reflejo lo más preciso posible de los modelos conceptuales. Para conseguir este fin, la traducción de los modelos conceptuales en el modelo de la base de datos se ha llevado a cabo empleando la metodología de modelado IDEF1X.

IDEF1X es una metodología que se ha empleado para el modelado de datos desde 1985, de una manera que resulte consistente, predecible, y genérica. En el lenguaje

empleado para la definición de la base de datos (BD), se especifican tanto la parte gráfica como la sintáctica de la estructura de la base de datos. Se trata de una herramienta estándar, cuyo uso en el modelado de base de datos está bastante extendido. Los diagramas IDEF1X, que encontraremos a lo largo del capítulo 4 de descripción de la BD, se caracterizan por los siguientes elementos principales:

- Entidad: Representan un conjunto de instancias de datos similares que tienen atributos y características en común. Se representan gráficamente en forma de rectángulos, y están divididas en dos partes. Por un lado, tenemos las entidades independientes, las cuales se representan con las esquinas en forma de pico, y son las que tienen todos los atributos definidos y no dependen de otra entidad para su definición. Por otro lado están las entidades dependientes, representadas con las esquinas redondeadas, son las que para ser totalmente definidas necesitan atributos de otras entidades, y por ello necesita relacionarse con otras entidades.
- Atributo: Representan las características distintivas y propias de una entidad, de la cual nos proporciona información. Para cada instancia de la entidad, la combinación de atributos de esa entidad toma valores distintos; pudiéndose así distinguir una instancia de otra.
- Clave: Está formada por uno o más atributos, y sirve para identificar de forma unívoca cada instancia de una entidad, así como para relacionar entidades. Cabe destacar tres tipos de clave:
  - Clave Primaria: Atributo o grupo de atributos que identifican unívocamente una instancia de una entidad. Aparece en la mitad superior de la entidad.
  - Clave Alterna: Atributo o grupo de atributos que identifican unívocamente cada instancia, pero que no forman parte de la clave primaria.
  - Clave Foránea: Una clave primaria de una entidad padre que es aportada a una entidad hijo, a través de una relación (Foreign Key, denotada con FK).
- Relación: Se usan para representar la asociación y vínculo entre dos entidades o instancias de la misma entidad. Para cuantificar el tipo de relación se emplea la *cardinalidad*, la cual especifica cuántas instancias de la entidad hijo pueden existir para cada instancia de la entidad padre. La cardinalidad se expresa mediante símbolos a ambos lados de la línea de relación, o mediante números. Las relaciones se dividen en dos grandes grupos:
  - Relación de conexión: Expresa cómo dos entidades se relacionan, y se representa mediante una línea. La relación puede der identificadora, en cuyo caso la línea será continúa y la clave principal de la entidad padre pasa a formar parte de la clave principal del hijo; o relación no-identificadora, donde la línea será discontinua y se da cuando la clave principal de la entidad padre no pasa a formar parte de la clave principal de la entidad hijo sino que forma parte de la clave alterna, apareciendo en la mitad inferior de la entidad.
  - Relación de categorización: Es usada para representar estructuras en las cuales una entidad es un tipo de categoría de otra entidad, permitiendo expresar una clasificación. Una instancia de la entidad genérica puede ser asociada con sólo una instancia de una categoría. Se representa con una línea y la ayuda de un círculo blanco. Si el círculo blanco tiene debajo dos líneas horizontales, refleja que la clasificación es completa, en caso de incorporar sólo una línea horizontal significa que la clasificación es incompleta, pudiendo haber más categorías. La cardinalidad no se representa, ya que siempre es cero o uno.

2.4.2 MICROSOFT ACCESS

Una vez explicada la metodología IDEF1X empleada para diseñar la base datos, acorde a los modelos conceptuales, en este apartado se presenta una introducción del sistema de gestión de base de datos (SGBD) relacionales escogido para la implementación de la BD, Microsoft Access.

La herramienta de Microsoft Access permite la implementación de la base de datos, modelada mediante IDEF1X, puesto que se trata de una herramienta que se emplea como sistema de gestión de base de datos relacionales, como es el caso de nuestra BD. Se ha seleccionado Access por tratarse de una herramienta estándar, de uso universal y muy extendido, y porque permite un acceso constante desde la herramienta encargada de la interface con el usuario, Visual Basic. De esta manera, desde Visual Basic se puede acceder a la base de datos de Access para insertar, borrar, y modificar datos, así como para realizar consultas. La potencia de la herramienta Access aporta las funcionalidades necesarias para la implementación de una base de datos relacional y compleja como la que requiere el actual proyecto.

Para la implementación de la base de datos en Access, se pude realizar una traducción literal de los modelos realizados con IDEF1X, ya que los elementos representados en un diagrama IDEF1X son los mismos que requiere Access. La BD en Access está formada por tablas, que equivalen a las distintas entidades del diagrama IDEF1X, y las relaciones entre ellas de acuerdo a lo expresado en el diagrama. A su vez, las tablas están formadas por tres elementos principales [4]:

- Campo: Es el equivalente a los atributos en el lenguaje IDEF1X, y definen las características o propiedades de la tabla que representan. Es cada una de las columnas de la tabla.
- Registro: Una tabla está compuesta por filas o registros donde se almacena información referida a una misma entidad particular. Se corresponde con la instancia de una entidad en el diagrama IDEF1X.
- Dato: Es la intersección entre un campo y un registro.

Por último, destacar que cada tabla está también compuesta por una clave primaria, que identifica unívocamente cada registro de la tabla, y una clave secundaria, compuesta por todos los campos que no pertenecen a la clave primaria. Para un mayor aprendizaje de la herramienta, se ha hecho uso de ar

2.4.3 UML

Una vez explicadas las herramientas empleadas para el diseño e implementación de la base de datos, en este apartado se explica la herramienta UML de modelado que se ha utilizado para el diseño de los modelos conceptuales entorno a los que se ha construido todo el proyecto.

El Lenguaje Unificado de Modelado (UML, Unified Modeling Language), desarrollado por la organización OMG (Object Management Group), es un popular lenguaje de modelado de sistemas de software. Se trata de un lenguaje gráfico para construir, documentar, visualizar, y especificar un sistema de software, incluyendo su estructura y diseño. Posee la riqueza y la potencia suficiente para crear un modelo del

sistema, pudiendo modelar los procesos de negocio y sistemas no software. La elección de la herramienta de modelado UML, para el diseño y construcción de los modelos conceptuales del proyecto, se debe por un lado a las altas prestaciones y posibilidades que la herramienta ofrece en el modelado de procesos de negocio, ligada a su orientación a los sistemas, y por otro lado debido al carácter estándar de lenguaje unificador con el que el UML fue creado y que le ha llevado a una amplia difusión de su uso y aceptación. La herramienta ofrece la posibilidad de emplear varios tipos diferentes de diagramas, sin embargo, en nuestro proyecto nos hemos centrado en el uso de dos tipos, diagrama de clases y diagrama de estados.

Los diagramas de clases son los diagramas fundamentalmente empleados en el proyecto. Representan las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Nos sirven para visualizar las relaciones entre las clases involucradas en el sistema. En un diagrama de clases se puede distinguir principalmente dos elementos: clases y relaciones [5].

- Clases: La clase es la unidad básica que contiene toda la información de un objeto a través del cual podemos modelar el entorno en estudio, y se representa por medio de un rectángulo que puede poseer tres divisiones: En la parte superior figurará el nombre de la clase, la parte intermedia contiene los atributos que caracterizan a la clase, y la parte inferior presenta los métodos u operaciones que representan la forma en la que el objeto interactúa con el entorno. Sólo es imprescindible la parte superior en la que se especifica el nombre, las otras dos sólo aparecerán si la clase lo requiere.
- Relaciones: Cabe destacar dos relaciones fundamentales, como son la relación de herencia (Generalización) y de asociación, aunque también emplearemos otras relaciones como la de composición. La relación de herencia indica que una clase (clase derivada) hereda los métodos y atributos especificados por una clase (clase base), por lo cual una clase derivada, además de tener sus propios métodos y atributos, podrá acceder a las características y atributos visibles de su clase base. La clase derivada es un subtipo de la clase base, y la relación se representa por una línea precedida de un triángulo apuntando a la clase base. Por otro lado, la relación de asociación permite asociar objetos que colaboran entre sí y que están en el mismo nivel. Adicionalmente, las relaciones pueden ir acompañadas de una multiplicidad, la cual determina cuántos objetos de cada tipo intervienen en la relación; se expresa a ambos extremos de la línea con el número correspondiente o con un asterisco, que significa que la multiplicidad puede ser cualquiera. Además, puede aparecer junto a la clase el rol con el que ésta participa de una relación.

El segundo tipo de diagrama UML empleado es el Diagrama de Estados. El diagrama de estados sirve para describir el comportamiento normal de un sistema, representando los diferentes estados por los que puede pasar un objeto que pertenece al sistema. Así mismo, incorpora los eventos, sucesos significativos, que dan lugar a cada tipo de cambio de estado, o a un momento significativo en la evolución del sistema. En la definición de dichos eventos, el diagrama muestra los valores de las condiciones que han de darse para que cada evento suceda. El punto de comienzo del diagrama se denota con un círculo negro. Cada estado posible, se representa con un rectángulo de esquinas circulares y el nombre del estado en el interior. El paso de un estado a otro se representa por flechas, que indican la dirección del cambio, y la definición del evento que puede

dar lugar al cambio junto a la flecha. El punto final del diagrama viene representado por un círculo negro en el interior de una circunferencia concéntrica.

2.4.4 VISUAL BASIC

Después de describir las herramientas empleadas para el diseño e implementación de la base de datos, y la herramienta de diseño de los modelos conceptuales, falta la herramienta que sirve como interface al usuario, y que le permite la definición flexible, rápida e intuitiva y resolución de los problemas de asignación de infraestructuras, a los cuales el proyecto ha sido enfocado, a través de los diferentes módulos requeridos. Esta herramienta es Visual Basic, y en este apartado se presenta una breve descripción de la misma.

Se ha empleado Visual Basic como interfaz gráfica con el usuario, y junto con la Base de Datos implementada en Microsoft Access, conforma la parte software del proyecto. Visual Basic es un lenguaje de tercera generación guiado por eventos. Es una herramienta de desarrollo que permite crear aplicaciones gráficas de forma rápida y muy sencilla. Se trata de un lenguaje y entorno de programación que puede ser aprendido de forma rápida por programadores inexpertos, característica que unido a su gran versatilidad, su uso extendido, la gran potencia como interfaz gráfica, y la posibilidad de coordinar la aplicación con una base de datos, han sido algunos de los criterios fundamentales para la elección de esta herramienta para la implementación del proyecto. Se han empleado fundamentalmente tres libros para adquirir el dominio necesario que requería el proyecto [1], [10], y [11].

La configuración de la parte visual del programa, básicamente consiste en crear ventanas (formularios) y añadir sobre ellas los controles que queramos utilizar, posicionándolos en la posición deseada. Cada objeto, ya sean controles o formularios, tiene una serie de propiedades y métodos que podremos manipular por código o sin la necesidad de código en muchas ocasiones. Este código se ejecutará cuando se produzca un suceso/evento determinado. Mientras ese suceso no se produzca, el código permanecerá inactivo. Algunos ejemplos de eventos pueden ser el click sobre un botón del formulario, el cierre del formulario, o el cambio en la elección entre una lista permitida de opciones. Los distintos tipos de sucesos pueden ser causados por el usuario, por el sistema (como el transcurso de un determinado periodo de tiempo), o por el código de forma indirecta. Cada evento suele estar ligado a unas líneas de código, aunque puede haber eventos sin código asociado. Este código se compone de instrucciones que se ejecutan durante el uso de la aplicación si sucede el evento correspondiente.

Por último, destacar que se ha tratado de llevar a cabo una programación modular, ya que al tratarse de un lenguaje guiado por eventos, y en general éstos no se encuentran relacionados por ninguna secuencia de ocurrencia, es necesario ésta técnica para aumentar la eficiencia y flexibilidad de la programación. Visual Basic incorpora la característica importante de poder asociar atributos a los controles, de manera que se puedan desactivar eventos asociados a dichos controles hasta que no se desencadenen eventos previos. Alguno de estos atributos son la propiedad Visible (visible) o Enabled (habilitado), que en caso de tomar valor False no permite hacer click sobre el control asociado.

CAPÍTULO 3

MODELOS CONCEPTUALES

Tras haber definido todos los aspectos que caracterizan el entorno del proyecto; tanto el punto de partida de planteamiento del proyecto, como los objetivos, los antecedentes, y el marco conceptual, este capítulo supone el núcleo del proyecto, ya que consta de todos los modelos conceptuales definitivos, adoptados tras el proceso de investigación y desarrollo. Se puede considerar el núcleo ya que en torno a dichos modelos se construye el resto del proyecto, es decir, tanto la base de datos como la aplicación que sirve de interfaz de usuario.

En este capítulo, se pretende describir de forma detallada todos los modelos conceptuales que conforman el proyecto, con ayuda de los diagramas UML y apoyándonos en las diferencias y similitudes respecto a los modelos de partida expuestos en el capítulo anterior. De esta manera, se trata de explicar todos los detalles y matices que nos han llevado a esta solución teórica.

Como se ha planteado en el anterior capítulo, correspondiente al marco teórico, el enfoque adoptado es de un modelo genérico jerárquico, y se compone de cinco niveles de abstracción. Por lo tanto, la estructura de este capítulo estará formada por un apartado introductorio, que incluye un ejemplo, y un apartado dedicado a cada nivel jerárquico, empezando por el nivel más abstracto, nivel de metamodelo de proceso, hasta el nivel más concreto, como es el nivel de ejecución (simulación).

3.1 INTRODUCCIÓN

En este breve apartado se incluye una introducción a los modelos conceptuales desarrollados, y se incluye un ejemplo que ayudará a entender mejor los apartados sucesivos.

Siguiendo el enfoque jerárquico en el modelado conceptual del presente proyecto, ha resultado necesario añadir un nivel de abstracción más respecto al modelo de partida. Este nuevo nivel es el correspondiente al conjunto de ejecuciones del proceso de negocio (Business Process Execution Set), y se sitúa entre la instancia de proceso de negocio y el nivel de ejecución. La necesidad de este nuevo nivel, así como su explicación detallada se presenta en el apartado dedicado a dicho nivel jerárquico.

En resumen, nuestro enfoque consta de los cinco niveles/capas siguientes, de mayor a menor nivel de abstracción de abstracción: Metamodelo de Proceso de Negocio (Business Process Metamodel), Modelo de Proceso de Negocio (Business Process Model), Instancia de Proceso de Negocio (Business Process Instance), Conjunto de Ejecuciones de Proceso de Negocio (Business Process Execution Set), y Ejecución de Proceso de Negocio (Business Process Execution).

Puesto que nuestra finalidad es la resolución de problemas en los que un cliente accede a través de un canal para reservar una infraestructura, en la figura 3-1 se presentan los cinco niveles jerárquicos acompañados de un ejemplo, que se presentó en el apartado 2.1.2, de reserva de habitaciones de hoteles y en el que se presenta a modo de ejemplo la definición de los clientes en los distintos niveles, ayudando así a tener una visión general del nivel de abstracción de cada nivel antes de profundizar en cada uno de ellos.

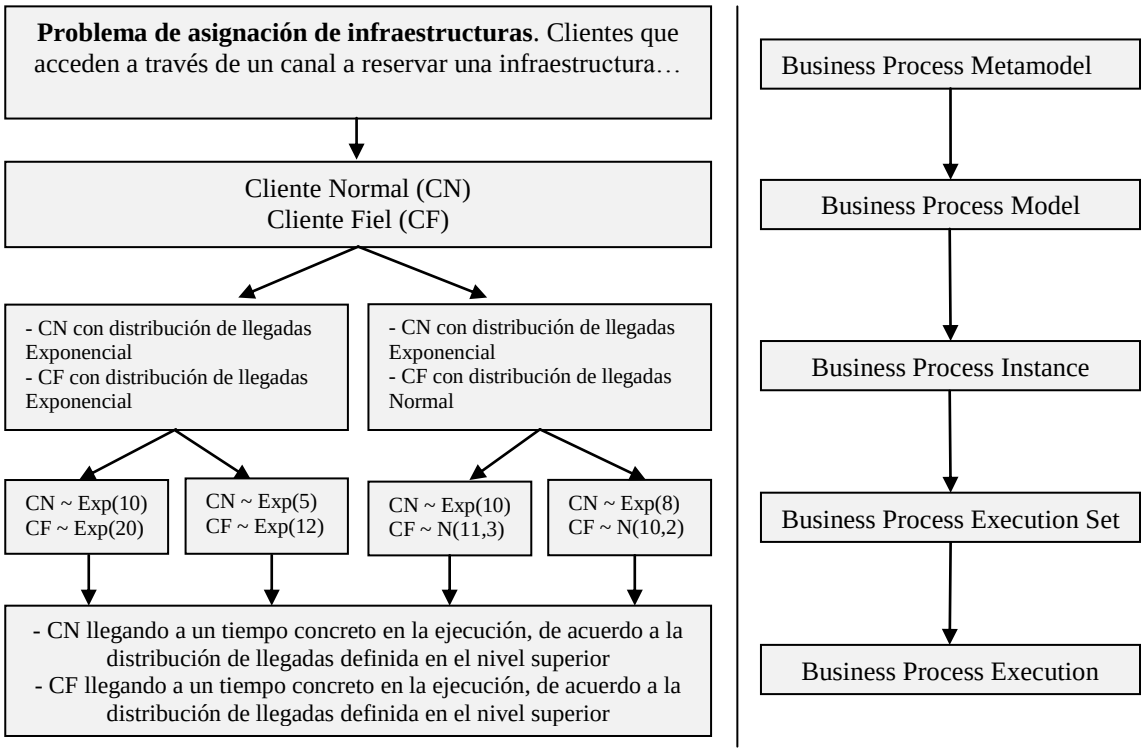


Figura 3-01. Niveles jerárquicos del modelado genérico y ejemplo

Cada uno de los niveles jerárquicos, que se explicarán en los siguientes apartados, vendrá acompañado de ejemplos para una mejor comprensión de los mismos, y en ocasiones se citará el ejemplo expuesto en la figura anterior (figura 3-1).

3.2 METAMODELO DE PROCESO DE NEGOCIO

En este apartado se describe el modelo conceptual perteneciente al primer nivel jerárquico, siendo el de mayor nivel de abstracción, llamado nivel de Metamodelo de Proceso de Negocio (Business Process Metamodel). Se destacarán también las diferencias respecto al modelo de partida.

En este nivel de Metamodelo, al tratarse del nivel de mayor abstracción del modelo, sólo se definen los elementos básicos que componen un problema de asignación de infraestructuras, así como la relación que deben mantener entre ellos; para que mediante un proceso de instanciación puedan definirse los posibles distintos procesos de negocio, correspondientes al siguiente nivel jerárquico. Los elementos básicos ya se han definido en el apartado de 2.1.1, correspondiente al metamodelo del modelo teórico de partida, y son los siguientes: **Segmento de Cliente** (Customer Segment), **Canal** (Channel), **Infraestructura** (Infrastructure), **Acceso a una Infraestructura** (Infrastructure Access), **Valor** (Value), **Tiempo** (Time), y **Asignación** (Allocation).

Atendiendo al diagrama UML de la figura 3-2, y partiendo de la parte superior, cada combinación de un Segmento de Cliente que accede a una Infraestructura a través de un Canal, conforma el elemento básico Acceso a una Infraestructura. En este nivel no hay nivel de restricción respecto a estas combinaciones, es decir, como denotan los asteriscos negros (multiplicidad cualquiera) del diagrama UML de la figura 3-2, la

combinación puede darse entre cualquiera de los tres elementos básicos mencionados. Cada Acceso a una Infraestructura vendrá definido por un Valor asociado, que en este nivel puede ser cualquiera, y que tendrá validez en un periodo de tiempo marcado por un tiempo inicial (Time Start) y un tiempo final (Time End) teniendo en cuenta las restricciones (Constraint) que se deben cumplir entre dichos instantes de tiempo. Cada combinación de los elementos Acceso a una infraestructura que recibe un valor asociado entre un tiempo inicial y un tiempo final, conforman el elemento básico asignación. Por lo tanto, en esta capa jerárquica, el metamodelo propuesto se compone de la definición de problemas de asignación de infraestructuras de una manera genérica, ya que ningún elemento básico toma valor, y las relaciones sólo reflejan los vínculos necesarios que se tendrán que definir en el siguiente nivel.

Finalmente, el diagrama concluye denotando que cada **Modelo de Asignación a Infraestructuras** (Infrastructure Allocation Model) está compuesto por un conjunto de asignaciones, así como de los otros 6 elementos básicos, como veremos en el siguiente nivel.

El Metamodelo de Proceso de Negocio que ha resultado del proyecto actual, presenta pocas diferencias respecto al metamodelo de partida reflejado en la figura 2-2 del capítulo 2. Las diferencias presentes se plasman en la clase Modelo de Asignación a Infraestructuras, que en el modelo de partida correspondía a Modelo de Proceso. Esta diferencia se explicará en detalla en el siguiente apartado, pero en resumen se debe a que en el modelo actual tanto un modelo de proceso de negocio como una instancias de modelo de proceso de negocio comparten las mismas características en cuanto a la definición de los elementos básicos del modelo y las relaciones entre ellos. Tanto un modelo de proceso de negocio como una instancia de un proceso de negocio son modelos de asignación a infraestructuras (Infrastructure Allocation Model), diferenciándose en el nivel de generalidad.

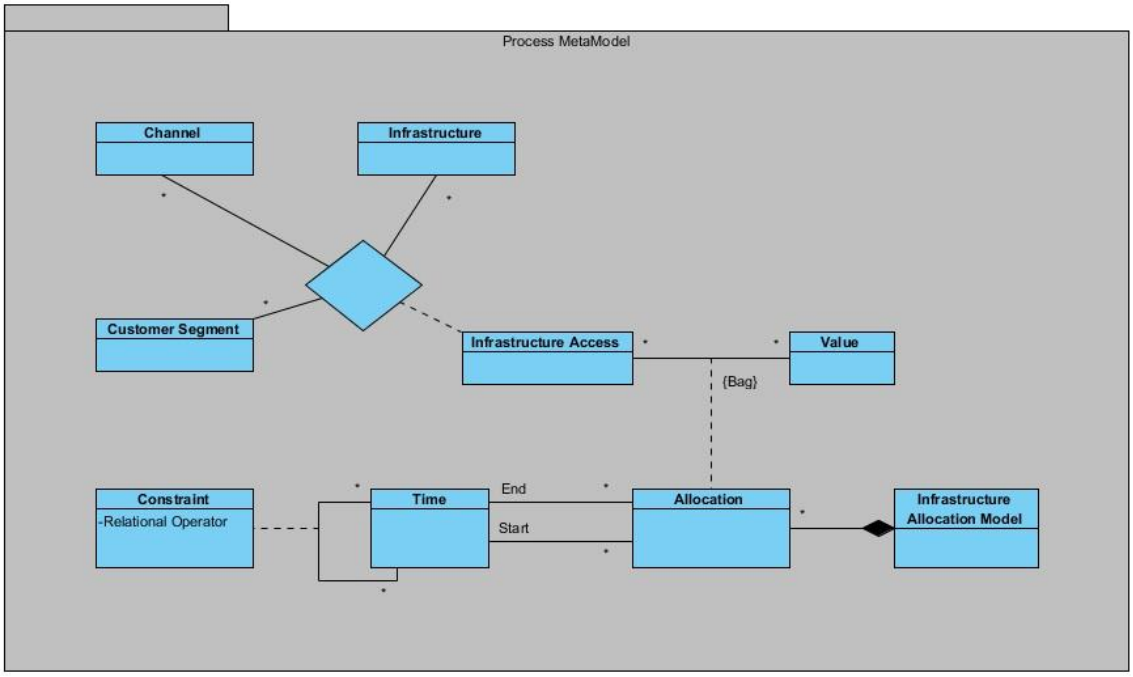


Figura 3-02. Diagrama de clase en UML del Process Metamodel en el diseño conceptual

3.3 MODELO DE PROCESO DE NEGOCIO

En este apartado pasamos al segundo nivel jerárquico del enfoque, y se explicará detalladamente el modelo de definición de procesos de negocio basándonos en los dos diagramas UML principales que definen el modelo conceptual de esta capa, las figuras 3-3 y 3-4. Puesto que tiene relación directa con los niveles superior e inferior, se explicará la relación con éstos y los matices más importantes.

En esta capa, ha sido necesario desarrollar unos modelos conceptuales nuevos respecto a los modelos de partida. Esto es porque durante el desarrollo del proyecto se buscó un modelado unificado de modelo de proceso e instancia de proceso. Este modelo unificado, nos posibilitó la implementación de una base de datos también unificada y que estuviera en línea con los modelos conceptuales.

La mejor manera de explicar los cambios adoptados es la descripción del diagrama UML mostrado en la figura 3-3. En esta figura, empezando por la parte superior, vemos que tanto un **modelo de proceso** de asignación de infraestructuras (Infrastructure Allocation Process Model) como una **instancia del proceso** de asignación de infraestructuras (Infrastructure Allocation Process Instance) son modelos de asignación a infraestructuras, como expresa la relación de generalización. Con esto, tanto los modelos de proceso de negocio como las instancias de los procesos de negocio son subtipos de modelos de asignación a infraestructuras, heredando sus características, y convirtiéndose así en "hermanos" de un "padre" que es la clase modelo de asignación a infraestructuras.

La gran diferencia entre un modelo de proceso y una instancia de proceso reside en el grado de generalidad, ya que la instancia por tratarse de un nivel más concreto (menos abstracto), incorporará características, como atributos y métodos, que hacen la instancia más concreta respecto al modelo de proceso. Además, como muestra la relación entre ambos, toda instancia de proceso de negocio tiene que pertenecer necesariamente a un modelo de proceso de negocio.

Por otro lado, tenemos siete entidades de modelado de asignación de infraestructuras (Infrastructure Allocation Modeling Entity), que corresponden con los siete elementos básicos descritos en apartados anteriores y en el metamodelo. Lo que expresa el diagrama UML es que todo modelo de asignación de infraestructuras debe tener los siete elementos/clases básicos, siendo cada uno una **entidad del modelo** de asignación de infraestructuras (IA Model Entity). Por lo tanto, las siete entidades del modelo son: cliente, canal, infraestructura, acceso a una infraestructura, valor, tiempo, y asignación. Como refleja la relación de generalización que apunta a IA Model Entity (Entidad del modelo), cada una de las entidades del modelo de proceso (Process Model Entity) es un subtipo de una entidad del modelo de asignación de infraestructuras, ocurriendo lo mismo con cada entidad de una instancia de proceso (Process Instance Entity).

A su vez, puesto que cada modelo de proceso de negocio y cada instancia de proceso son subtipos de modelo de asignación, contarán necesariamente con las siete entidades del modelo de asignación de infraestructuras, que son las siete entidades del modelo de proceso y las siete entidades de la instancia de proceso respectivamente.

En resumen, cada modelo de proceso de asignación de infraestructuras y cada instancia de proceso de asignación de infraestructuras son subtipos de modelo de asignación de infraestructuras. Al ser subtipos y heredar sus características, como cada modelo de asignación de infraestructuras cuenta necesariamente con las siete entidades del modelo, todo modelo de proceso contará con las siete entidades (Entidad de modelo de proceso) y toda instancia de proceso contará con siete entidades (Entidad de la instancia de proceso). Siendo las entidades del modelo de proceso y las entidades de la instancia de proceso subtipos de entidad de modelo de asignación de infraestructuras.

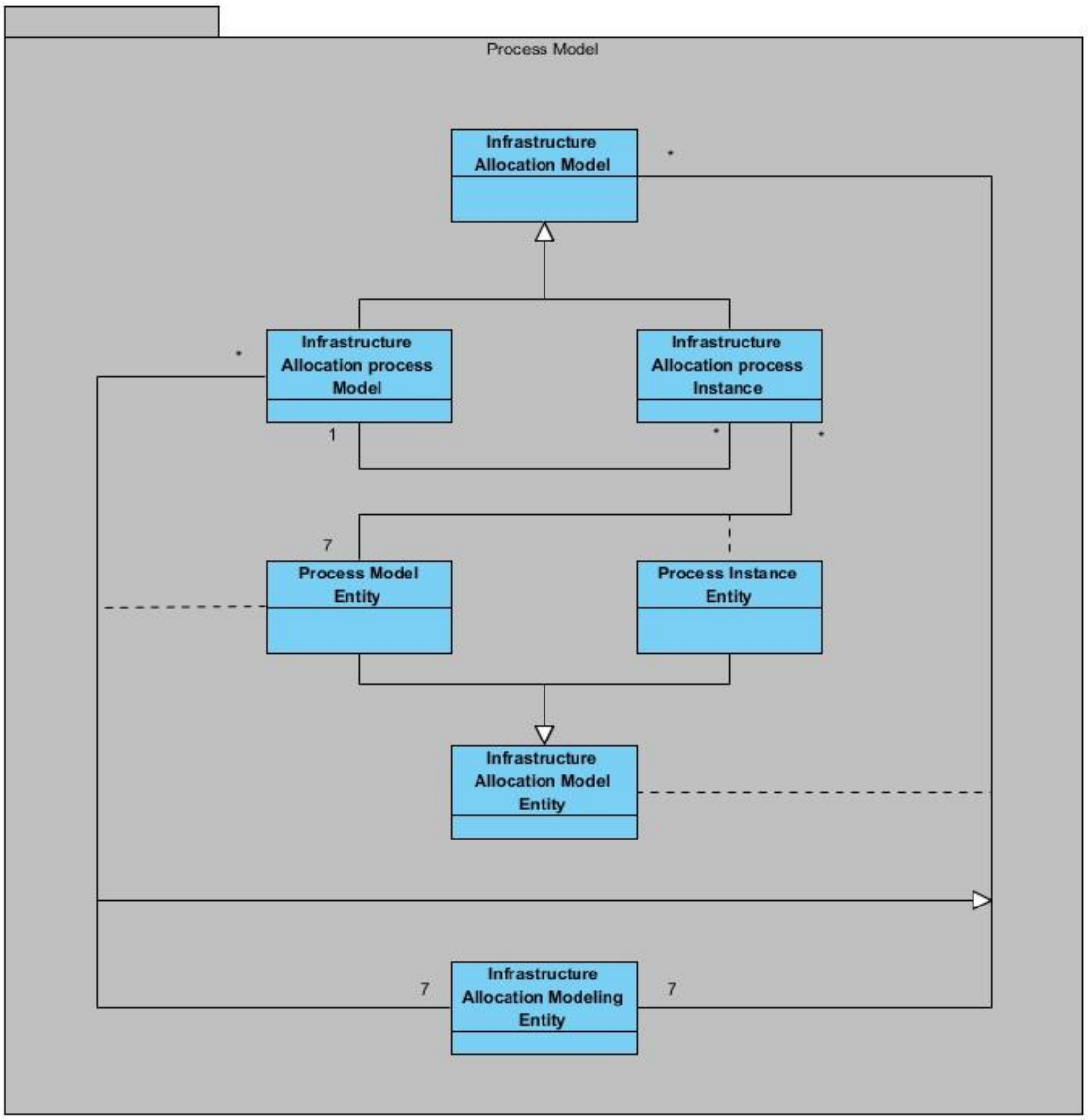


Figura 3-03. Diagrama de clase en UML del Process Model en el diseño conceptual

La clase modelo de asignación de infraestructuras es la unión entre el metamodelo, explicado en el apartado anterior, y el modelo de proceso como se puede observar en las figuras 3-2 y 3-3. Como hemos explicado en el apartado anterior, cada modelo de asignación de infraestructuras está compuesto por asignaciones, así como los seis elementos básicos restantes. Respecto al nivel de proceso de negocio, los subtipos de un modelo de asignación de infraestructuras heredan esta característica, es decir cada modelo de proceso y cada instancia de proceso están compuestos por dichos elementos.

La figura 3-4 completa el modelo conceptual de proceso de negocio, y muestra las diferentes entidades de modelo de asignación a infraestructuras y los elementos de cada entidad. La clase de unión con el diagrama de la figura 3-3 es la entidad del modelo de asignación a infraestructuras (Infrastructure Allocation Model Entity).

Como hemos mencionado arriba, todo modelo de proceso y toda instancia de proceso tiene necesariamente las siete entidades básicas que conforman un modelo de asignación de infraestructuras. El modelo de la figura 3-4 muestra la clasificación completa y disjunta de las entidades de un modelo de asignación de infraestructuras. Por tratarse de una clasificación completa, toda entidad del modelo debe pertenecer a una y sólo una de las clases que se muestran en la clasificación, es decir, debe pertenecer a una de las siguientes entidades: Segmento de cliente, canal, infraestructura, acceso a infraestructura, tiempo, valor, o asignación.

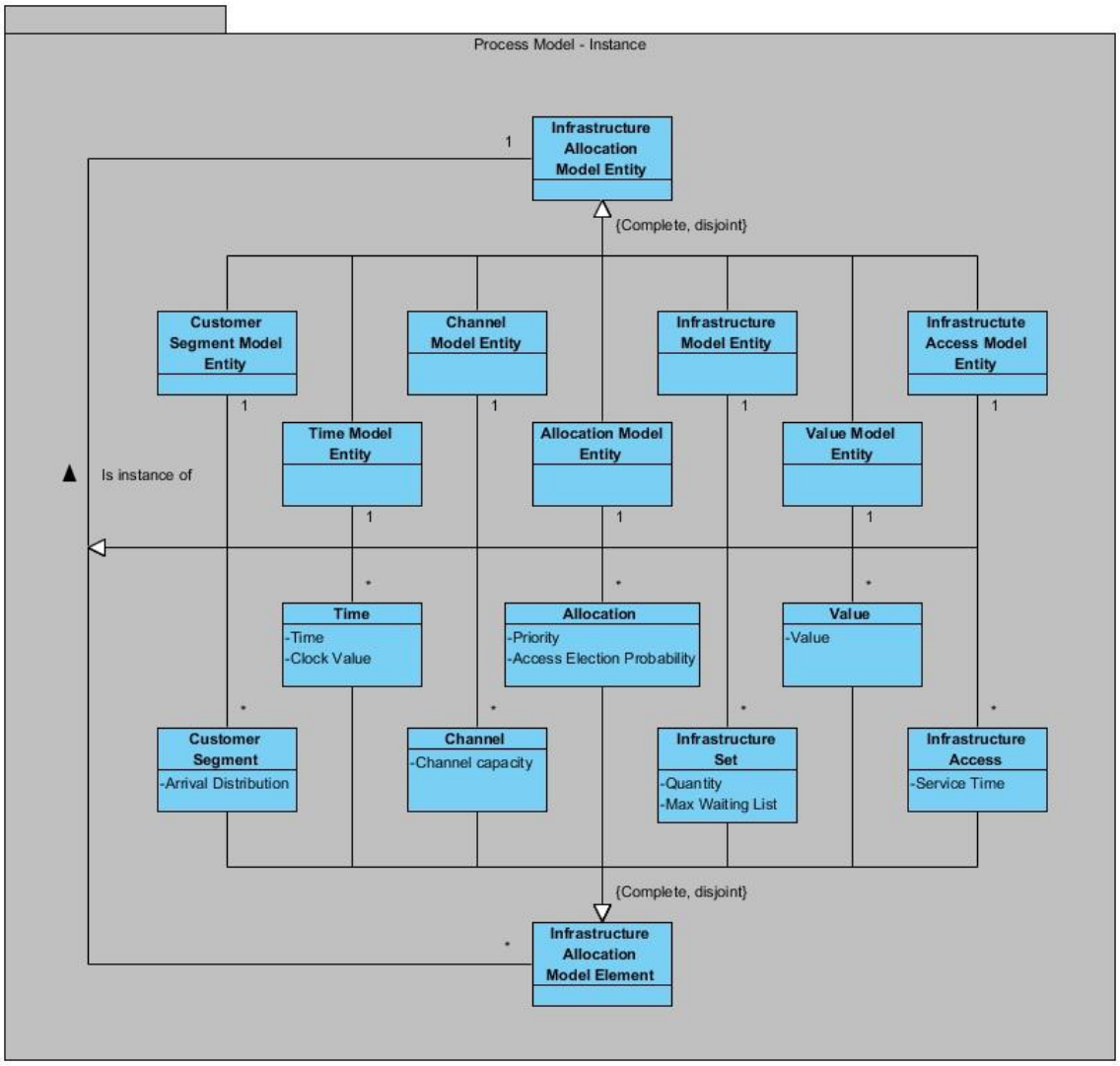


Figura 3-04. Diagrama de clase en UML de las entidades del modelo

Como muestra el gráfico 3-4, cada entidad de un modelo de asignación de infraestructuras puede tener multitud de elementos del modelo (Infrastructure Allocation Model Element), en una relación del tipo "Is instance of", ya explicada en el apartado 2.1.3, y que representa una relación de generalización de la instanciación, en la que el elemento del modelo (Model Element) es una especialización de la entidad del

modelo (Model Entity), que representa el "padre". Como se explicó, mediante esta relación, todos los "hijos" (Elementos del modelo) heredan las características del padre (Entidad del modelo).

Al igual que ocurre con las entidades del modelo, los elementos del modelo (Model Element) se pueden clasificar de manera completa y disjunta en: Segmento de cliente, canal, infraestructura, acceso a infraestructura, valor, tiempo, o asignación. Cada elemento puede pertenecer sólo a una de estas clases.

En resumen, existen siete entidades en un modelo de asignación de infraestructuras, las nombradas anteriormente, y cada una de estas entidades puede tener muchos elementos del modelo provenientes de dicha entidad, en una relación uno a mucho y de tipo generalización de la instanciación "Is instance of", por la cual los elementos del modelo heredan las características de las entidades del modelo. Así pues, la entidad del modelo segmento de cliente puede tener muchos elementos segmento de clientes, los cuales tendrán características más concretas que la entidad padre, y lo mismo ocurre con el resto de entidades del modelo.

Por ejemplo, algunas instancias del elemento del modelo Segmento de Cliente pueden ser: cliente mayor de 25 años, cliente fiel, cliente seriamente herido, etc. En el caso del elemento del modelo Canal, algunas instancias pueden ser: Ambulancia, recepción, internet, etc.

Por último, concluir que, como se ha comentado al inicio del apartado, el modelo que se ha desarrollado en este nivel es un modelo unificado, por lo que tanto un modelo de proceso de negocio como una instancia de proceso de negocio deben constar de las siete entidades del modelo (Infrastructure Allocation Model Entity). Asimismo, y como refleja el modelo de la figura 3-4, tanto un modelo de proceso como una instancia de proceso deben contener al menos un elemento del modelo de cada clase; es decir, al menos un elemento de la clase segmento de cliente, canal, infraestructura, acceso a una infraestructura, valor, tiempo, y asignación. Además, por tratarse de un modelo unificado, en el que el modelo de proceso y la instancia de proceso son "hermanos" aunque diferenciados en el grado de generalidad, los elementos del modelo (Model Element) presentes en un modelo de proceso de negocio, serán los mismos que los presentes en la instancia del proceso de negocio, aunque en ésta última con características más concretas por tratarse de un nivel menos abstracto. Los elementos del modelo son definidos en el nivel de modelo de proceso, aunque en el nivel de instancia de definen más propiedades de dichos elementos.

Aunque en todo modelo de proceso de negocio y por tanto en toda instancia de proceso de negocio debe definirse al menos un elemento del modelo de cada clase, generalmente se definirán varios elementos de cada clase. Cuantos más elementos del modelo se definan, más complejo será el modelo de proceso de negocio; pero al tratarse de un enfoque de modelado genérico, está diseñado para ser aplicable aunque aumente la complejidad de los procesos de negocio.

3.4 INSTANCIA DE PROCESO DE NEGOCIO

Tras la explicación del nivel jerárquico de modelo de proceso de negocio, el tercer y siguiente nivel es el nivel de la instancia del proceso de negocio. En este apartado se

pretende explicar detalladamente todos los aspectos importantes del modelo perteneciente al nivel de instancia de proceso de negocio. Para ello, este apartado constará de varias secciones que irán recorriendo las distintas partes que componen este nivel. Concretamente, el apartado comienza con una introducción, donde se explican los aspectos generales más importante de este nivel. El apartado consta de una primera sección donde se explican los modelos de atributos, la segunda sección consta de los modelos asociados a los estados y eventos, y por último una sección donde se describe el modulo de la función objetivo. Todas estas secciones pertenecen al nivel de instancia.

Como se ha explicado en el apartado anterior, tanto la figura 3-3 como la figura 3-4 son aplicables a este nivel de abstracción. Como se ha explicado, y como refleja la figura 3-3, se ha desarrollado un modelo conceptual unificado entre el modelo de proceso de negocio y la instancia de proceso de negocio, por el cual se considera que tanto un modelo de proceso (Process Model) como una instancia de proceso (Process Instance) son subtipos de un modelo de asignación de infraestructuras (Infrastructure Allocation Model). En este modelo, un modelo de proceso y una instancia de proceso son "hermanos", aunque se diferencian en el grado de generalidad, siendo el modelo de proceso más abstracto que la instancia de proceso, la cual es más concreta desde el punto de vista de que presenta elementos con más características definidas (más concretos). Además, toda instancia pertenece a un modelo de proceso, y presenta las mismas características que dicho modelo, incorporando nuevos atributos y métodos.

Como se ha explicado, toda instancia, al igual que todo modelo de proceso, consta de las siete entidades del modelo llamadas en este nivel entidades de la instancia de proceso (Process Instance Entity), así como de los elementos del modelo que son "hijos" de cada una de estas entidades y que son definidos en el nivel de modelo de proceso. Como es un modelo unificado, todos los matices explicados en el apartado anterior son directamente aplicables en este nivel.

Respecto al modelo de partida, al adoptar un modelo unificado, se han cambiado varios matices e implicaciones, aunque el concepto general no se ha modificado.

Dado que este nivel es un nivel más concreto que el correspondiente al modelo de proceso de negocio, las secciones que se detallan a continuación describen todos los aspectos que aportan más características a los elementos del modelo, disminuyendo el nivel de abstracción y avanzando en el proceso de instanciación. La siguiente sección está dedicada a la definición y propiedades de los atributos asociados a los elementos del modelo.

3.4.1 MÓDULO DE DEFINICIÓN DE ATRIBUTOS

Esta sección se enmarca dentro del modelo conceptual perteneciente al nivel de instancia de proceso. Se trata de un módulo que junto con el módulo de estados, que veremos en la siguiente sección, y el módulo de función objetivo, que veremos en la sección 3.4.3, aporta características a los elementos del modelo que los hace más concreto y disminuye el grado de abstracción/generalidad. Esta sección está dedicada a la definición de los atributos propios de cada entidad del modelo (En este nivel Entidades de la Instancia), así como la clasificación de éstos y las posibilidades que ofrecen de cara a la definición de instancias del proceso que nos ocupa. Para ello se explicará n los modelos UML y las implicaciones que supone cada uno de ellos.

Puesto que el proyecto está dirigido a resolver una amplia gama de problemas de asignación de infraestructuras, el usuario necesitará de atributos para definir la instancia de proceso y la buscada optimización del problema planteado. Los atributos permitirán al usuario definir los parámetros y variables necesarias para una correcta optimización del modelo de proceso planteado. Por lo tanto, los atributos permitirán al usuario, de forma flexible, adaptar la definición de la instancia de proceso al entorno específico real del proceso de negocio que se pretende optimizar, haciendo la herramienta DSS más potente y eficaz. Además, los atributos son necesarios para una buena caracterización de los eventos que pueden suceder, y una buena definición de las funciones objetivo, como veremos en las dos secciones siguientes.

Un atributo es una característica o propiedad que posibilita disminuir el grado de abstracción o generalidad de una instancia de proceso. Como explicaremos a continuación, tanto las entidades de la instancia (Instance Entity) como los elementos que derivan de éstas adquieren estas características proporcionadas por los atributos, lo que supone uno de los puntos que diferencia a una instancia de proceso de un modelo de proceso ya que la instancia presenta un nivel más concreto, con elementos y entidades más concretas. Además, como explicaremos a continuación, cada atributo se asocia con una entidad del modelo, de manera que es una característica que le es propia a la entidad de la instancia que es asociado, y por instanciación es propia de todos los elementos del modelo que derivan de dicha entidad; si bien en cada caso (cada elemento) esa característica o propiedad podrá adquirir un valor distinto o no tomar valor hasta el siguiente nivel de abstracción.

Los atributos permitirán al usuario definir los parámetros y las variables del problema de acuerdo a las necesidades requeridas, aportando flexibilidad y una aproximación más realista a la hora de definir el árbol de estados gobernado por los posibles eventos, y las funciones objetivo que se pretenden optimizar.

En la figura 3-5 se muestra el modelo correspondiente al módulo de atributos de la instancia de proceso, el cual representa la base que nos ayuda a explicar todos los matices importantes. En primer lugar, como se aprecia en la parte superior del diagrama y como ya hemos introducido, la definición de cada atributo de forma genérica va asociado necesariamente a una entidad del modelo (en este nivel entidad de la instancia); es decir, cada atributo definido de forma genérica pertenece necesariamente a una de las siete entidades del modelo, y esta puede considerarse la primera clasificación de los atributos. Esta definición genérica consiste en asociar el atributo a una entidad del modelo, atribuirle un nombre, y definir el tipo de atributo de acuerdo a la clasificación que se presenta en el diagrama. Aquí se presenta la segunda clasificación posible, puesto que la primera era la pertenencia a una entidad del modelo u a otra, y es que a la hora de definir un atributo de forma genérica, éste se clasifica de manera completa y disjunta en: Distribución estadística (Distribution), probabilidad (Probability), constante (Constant) o booleano (Boolean). En el caso de las distribuciones, se deberá indicar su tipo en el momento de definición del atributo, que se almacena en Type, que es el atributo de un atributo y se muestra en el diagrama UML de la figura 3-5.

El modelo permite elegir cualquier tipo de distribución, ya sea exponencial, normal, poisson, binomial, etc. Sin embargo, la gama de distribuciones entre las que se podrá escoger dependerá de la biblioteca de distribuciones estadísticas que presente la herramienta que se usará para la definición del modelo y su optimización. Respecto al segundo tipo de atributo mencionado, si el atributo es de tipo probabilidad, éste se

clasificará a su vez de manera completa en atributos de probabilidad única (Single Probability) o como atributos de probabilidad definida a trozos (Piecewise Probability). Este último tipo de atributo nos permite establecer distintos valores probabilísticos en función del intervalo en el que nos encontremos. Se explicará con detalle más adelante, pues consta de un modo particular a la hora de asignar valores a los parámetros que lo definen. El siguiente tipo de atributo es constante, y como su propio nombre indica se caracteriza por adquirir como valor una constante numérica. Por último, un atributo puede ser booleano o lógico, lo que significa que sólo puede ser verdadero o falso.

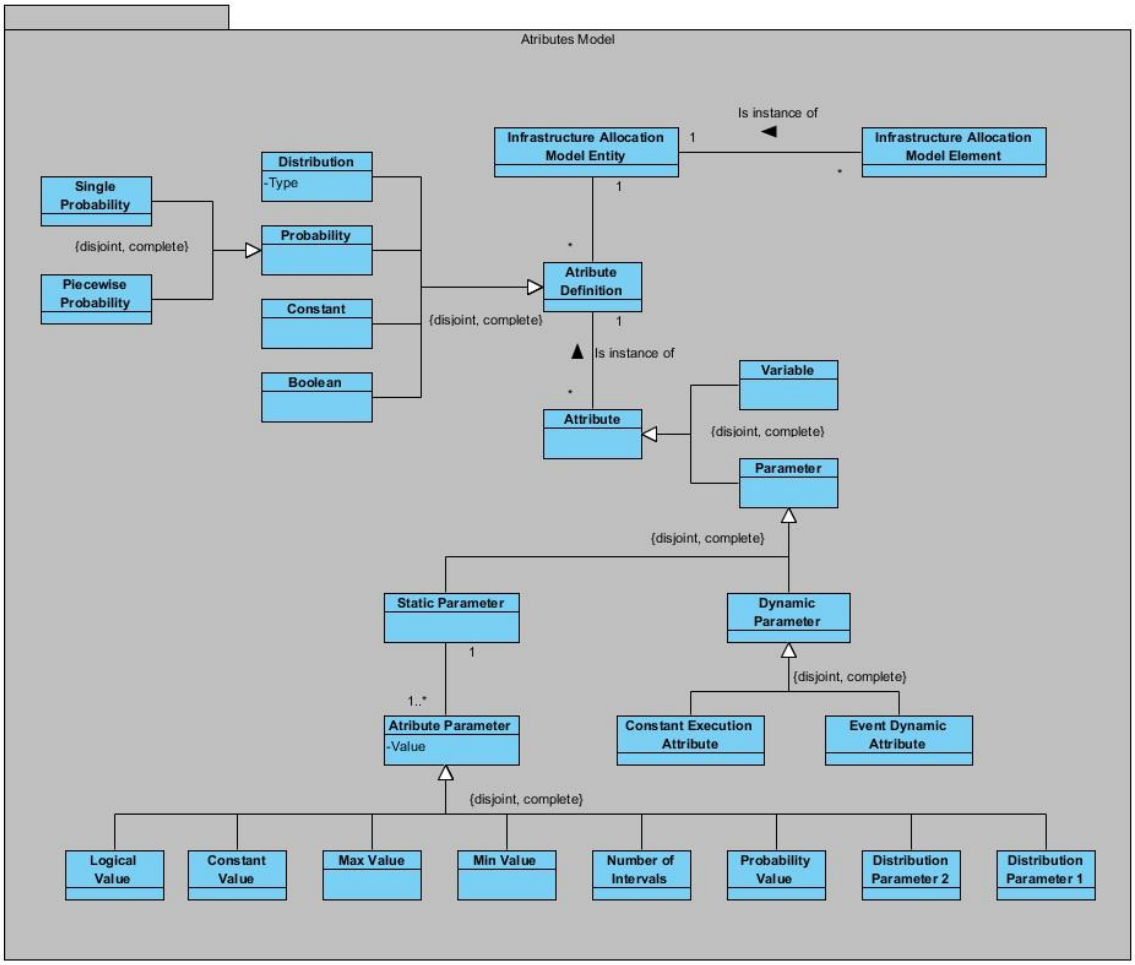


Figura 3-05. Diagrama de clase en UML correspondiente al Modelo de Atributos

Avanzando en el diagrama de la figura 3-5, vemos que de cada atributo definido de forma genérica (Attribute Definition) derivan mediante una relación de instanciación genérica "Is instance of" muchos atributos (Attribute). Mediante esta relación, ya explicada, los atributos heredan las características del padre (Attribute Definition), por lo que serán del mismo tipo que éste. Mediante esta relación de instanciación, se alcanza un nivel más concreto, y estos atributos estarán asociados a los elementos que derivan de las entidades de la instancia a la que a sido ligado la definición del atributo. Para aclarar este concepto se emplea el ejemplo del apartado introductoria del actual capítulo, y se muestra a continuación.

Por un lado tenemos un atributo llamado "Distribución de llegadas", de tipo distribución exponencial y asociado a la entidad de la instancia Segmento de Cliente. Por otro lado, de la entidad de la instancia Segmento de Cliente se dirivan mediante relación de instanciación los elementos "Cliente Fiel" y "Cliente Normal" (Definidos en

el modelo de proceso de negocio). Por lo tanto, habrá un atributo hijo de "Distribución de llegadas" asociado a "Cliente Fiel" y otro asociado a "Cliente Normal", de manera que cada uno pueda adquirir un valor diferente dependiendo del elemento al que va asociado. Es decir, que "Cliente Fiel" pueda tener una "Distribución de llegadas" que siga una exponencial de media 20, y "Cliente Normal" una exponencial de media 10.

Puesto que estos atributos (Attribute) son más concretos, contienen más características. Estos atributos, a su vez deben ser clasificados de forma completa y disjunta en parámetros (Parameter) o variables (Variable). Aquí se define qué atributos asociados a qué elementos del modelo van a funcionar como parámetro, y cuáles como variables y cuya optimización es el propósito del problema de acuerdo a unas funciones objetivo. Las variables, en la búsqueda del valor óptimo, tomarán los valores posibles en el siguiente nivel jerárquico, en el nivel de conjunto de ejecuciones (Execution Set). Cabe destacar que sólo podrán ser definidaos como variables aquellos atributos que sean de tipo constante (Constant) o booleano, puesto que los tipos distribución y probabilidad son asociados a inputs (Valores dados) del modelo, ya que toman valores aleatorios durante la ejecución y no se podría garantizar una correcta optimización. Sin embargo, como parámetros pueden ser definidos cualquier tipo de atributos.

Por su parte, los atributos definidos como parámetros, se clasifican de forma completa y disjunta en **parámetros estáticos** (Static Parameter) o **parámetros dinámicos** (Dynamic Parameter). Sobre esta clasificación, es necesario detallar las implicaciones que supone declarar el atributo como un tipo u otro, y por ello en los siguientes párrafos nos centraremos en describir cada una de estas dos clases.

1) *Parámetros estáticos*

Los atributos definidos como parámetros estáticos serán aquellos que tomarán un valor concreto en este nivel de instancia de proceso, y el cual se mantendrá invariable durante las sucesivas ejecuciones (simulaciones).

Como se muestra en el diagrama UML de la figura 3-5, cada atributo declarado parámetro estático tendrá al menos un parámetro de atributo (Attribute Parameter) que necesitará tomar un valor en este nivel de instancia, y asignado por el usuario. Dichos parámetros de atributo se clasifican de manera completa y disjunta de la siguiente manera:

1) *Logical Value*: Se trata de un valor lógico que sólo puede tomar los valores de verdadero (True) o falso (False). Este parámetro de atributo está asociado sólo a los atributos de tipo booleano (Boolean).
2) *Constant Value*: Valor numérico que determina el valor que adquieren los atributos clasificados como constantes (Constant). Asociado a este tipo de atributos.
3) *Distribution Parameter 1*: Relacionado con atributos del tipo distribución (Distribution). Es un parámetro numérico que define las características de la distribución a la que se ajusta el atributo. Por ejemplo la media de una distribución normal.
4) *Distribution Parameter 2*: Análogo al anterior parámetro de atributo. Por ejemplo para definir la mediana de una distribución normal.
5) *Probability value*: Valor probabilistico, por lo que debe estar comprendido entre 0 y 1. Este parámetro de atributo está asociado a los atributos de tipo probabilidad, tanto los de probabilidad única (Single Probability) como los de probabilidad

definida a trozos (Piecewise Probability), y en este último caso habrá uno para cada intervalo definido (Probabilidad distinta para cada intervalo).

6) *Max Value*: Valor numérico que indica cuál es el valor que marca el extremo superior del rango en el que se define un atributo de probabilidad a trozos.
7) *Min Value*: Valor numérico que indica cuál es el valor que marca el extremo inferior del rango en el que se define un atributo de probabilidad a trozos.
8) *Number of intervals*: Número entero que indica en cuántos intervalos distintos se divide el rango comprendido por un atributo del tipo probabilidad a trozos.

Los tipos de parámetros de atributos explicados se relacionan con los atributos correspondientes, indicados en la descripción de cada parámetro. De esta manera, los atributos de tipo distribución tendrán los parametros de atributo asociados Distribution Parameter 1 y Distribution Parameter 2, los de tipo probabilidad única (Single Probability) tendrán asociado un Probability Value, los de tipo constante vendrán acompañados de un Constant Value, y los de tipo booleano tendrán asociado un Logical Value.

En el caso de un atributo de tipo probabilidad a trozos requiere de más parámetros de atributo para la asignación de valores. Requerirá de un Max Value, un Min Value, un Number of intervals, y tantos Probability Value como intervalos tenga el rango, ya que cada Probability Value definirá la probabilidad en un intervalo diferente. Para su mejor entendimiento, en la figura 3-6 se presenta un ejemplo:

Min Value	Max Value	Number of intervals	Probability Value	Probability Value	Probability Value
0	150	3	0,3	0,6	0,9

Interval	Probability Value
0 - 50	0,3
50 - 100	0,6
100 - 150	0,9

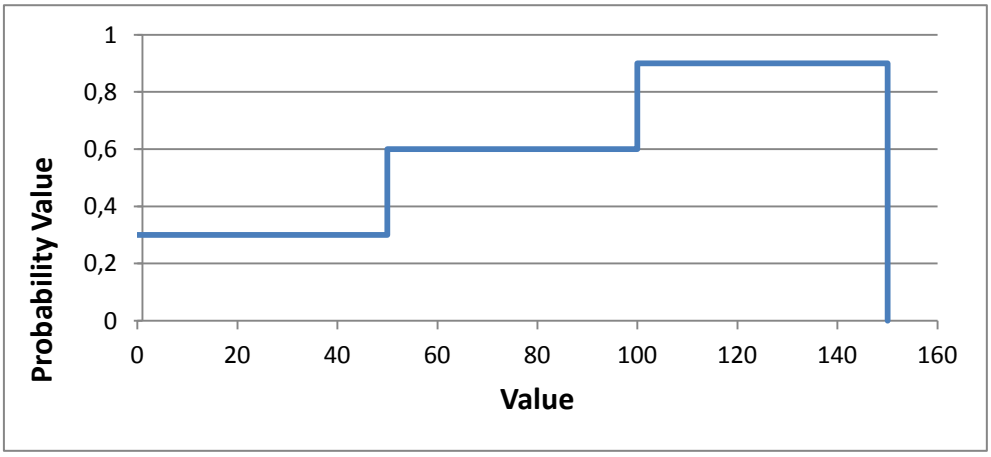


Figura 3-06. Parámetros de atributo y gráfico de Probabilidad definida a trozos

A la hora de emplear un atributo de este tipo, se debe combinar con otro atributo que determine el valor con el que se pueda evaluar la probabilidad dado dicho valor, es decir, un valor con que entrar en la gráfica y obtener la probabilidad correspondiente. El valor del atributo combinado sería el input, y la probabilidad el output.

Un ejemplo de atributo de tipo probabilidad a trozos es el atributo llamado “Value Based Decision”, que se explicará más adelante, y que supone uno de los atributos necesarios para cualquier modelo. Este atributo define la probabilidad de que un cliente

que pertenece a un determinado segmento de cliente, y que accede a una infraestructura a traves de un canal concreto, acepte la oferte en función del valor que ésta presente. Por lo tanto, la probabilidad dependerá del valor asociado a la asignación (Allocation), que tipicamente es el precio pero podría ser otro tipo de valor.

2) *Parámetros dinámicos*

Los atributos dinámicos, son aquellos que no adquieren ningún valor concreto en este nivel de instancia de proceso, y su valor puede cambiar para cada conjunto de ejecuciones. Como muestra el diagrama UML de la figura 3-5 mostrado anteriormente, todo atributo dinámico se clasifica de manera completa y disjunta en atributo dinámico de ejecución constante (Constant execution attribute) o atributo de evento dinámico (Event Dynamic Attribute). Cada una de estas clases contiene diferentes características, pero ambos se caracterizan en que pueden adquirir distinto valor para conjuntos de ejecuciones pertenecientes a la misma instancia de proceso.

a) Atributo dinámico de ejecución constante

Este tipo de atributo, al ser dinámico no adquirirá valos en el nivel de instancia, pero lo hará en el siguiente nivel jerárquico llamado "Execution Set" o conjunto de ejecuciones ("Snapshot"). Como se explicará en el siguente nivel jerárquico, y puesto que es nuevo respecto al modelo inicial de partida, una de las razones por las que surgió la necesidad de incluir este nuevo nivel fue para poder añadir un nivel que permitiera al modelo ser más concreto a través de la definición de los valores de los atributos dinámicos. De esta forma, los atributos dinámicos de ejecución constante son definidos en el nivel de instancia, donde se define el nombre del atributo, la entidad de la instancia a la que v asociado, y el tipo de atributo, pero no se define el valor que adoptará.

Por lo tanto, el valor concreto de estos atributos será definido en el nivel jerárquico de conjunto de ejecuciones, añadiendo un mayor grado de instanciación a la instancia de proceso de la que procede dicho conjunto de ejecuciones. Como explicaremos en el siguiente nivel, de una instancia de proceso pueden derivar muchos conjuntos de ejecuciones, mediante una relación de instanciación, de manera que cada conjunto de ejecuciones es hijo de la instancia, y por lo tanto hereda las propiedades y métodos. Al heredar las propiedades, hereda los atributos definidos como dinámicos de ejecución constante. Ha sido necesario considerar este nuevo nivel (Execution Set) pues reduce el grado de abstracción de la instancia de proceso mediante la definición de los valores de los atributos dinámicos.

Así pues, empleando el ejemplo mostrado en la figura 3-1 a modo de resumen, en el nivel de instancia se ha definido que el elemento del modelo "cliente normal" tenga asociado un atributo "distribución de llegadas" que se ajuste a una distribución exponencial. Mediante instanciación, en el nivel de "Execution Set", este atributo se hace más concreto al adquirir un valor de la media de la distribución de 10. Sin embargo, al tratarse de un atributo dinámico, en otro conjunto de ejecuciones, también perteneciente a la misma instancia de proceso, se podría definir un valor de la media de 12, dando origen a ejecución caracterizadas de forma diferente al caonjunto de ejecuciones con una media asociada de 10.

En resumen, es necesario el nivel jerárquico de "Conjunto de ejecuciones" (Execution Set), ya que en el nivel de instancia de proceso no se definen todas las

propiedades del modelo de proceso, y por lo tanto se pueden presentar muchos conjuntos de ejecuciones definidas de forma diferente (Atributos dinámicos adquieren un valor diferente) aunque pertenecientes a la misma instancia de proceso.

A diferencia de los atributos de evento dinámico, los atributos dinámicos de ejecución constante no pueden cambiar de valor a lo largo de una ejecución/simulación, sino que el valor que caracteriza al atributo será el mismo al inicio y al fin de la ejecución.

b) Atributo de evento dinámico

Los atributos de evento dinámico suponen una potente incorporación a las posibilidades de la herramienta y a la robustez de los modelos. Gracias a la incorporación de este tipo de atributos, es posible modelar procesos de negocio más complejos y de forma más realista y más ajustada al entorno específico del proceso. Se trata de atributos que ayudan tanto a una mejor definición del proceso de negocio a través de los diferentes niveles como una ejecución/simulación más eficiente y cercana a la realidad.

Los atributos de evento dinámico se caracterizan porque su valor puede ir cambiando a lo largo de cada ejecución. Dado un valor inicial con el que comenzará la ejecución, cualquier evento que suceda durante la simulación puede desencadenar un cambio en el valor del atributo. Ya que se trata de un tipo de atributo dinámico, los atributos de evento dinámico no adquieren valor en el nivel de instancia de proceso, y en este nivel sólo se define el tipo de atributo que es, a qué entidad de la instancia va asociado, a qué elemento del modelo va asociado, y la caracterización del atributo, que explicaremos a continuaión. Será en el nivel siguiente, nivel de conjunto de ejecuciones, donde se define el valor inicial que adquirirá el atributo al inicio de la ejecución, pudiendo cambiar a lo largo de la misma.

Cada atributo de evento dinámico se asocia en la instancia de proceso a un elemento del modelo, y se ha de definir qué eventos producrán cambios en el valor del atributo, qué cambios en el valor producirá ese evento, y que elemento del modelo provocará el evento que desencadena el cambio. Este proceso de definición de las condiciones mencionadas se llama caracterización del atributo de evento dinámico (Event dynamic attribute characterization). En la figura 3-7, se presenta el modelo UML correspondiente a la caracterización de atributos de evento dinámicos. En la figura se puede apreciar que la combinación de un atributo de evento dinámico, con un tipo de evento, y un elemento de la clase asignación (Allocation), da lugar a una caracterización de dicho atributo. De esta manera, cada atributo de evento asociado a un elemento del modelo puede cambiar de valor cuando un elemento asignación produce la ocurrencia de un evento.

La multiplicidad mostrada en el diagrama refleja que cualquier atributo de evento puede cambiar de valor al suceder cualquier evento, y producido por cualquier elemento asignación, o en el caso de ser un evento de tiempo, no estará caracterizado por ningun elemento asignación. Como veremos en el siguiente apartado, hay dos tipos de evento, un cambio de estado o un evento de tiempo que supone un hito en la ejecución. Por lo tanto, un atributo de evento cambiará de valor cuando un elemento asignación sufra un determinado cambio de estado, o cuando se produzca un evento de tiempo.

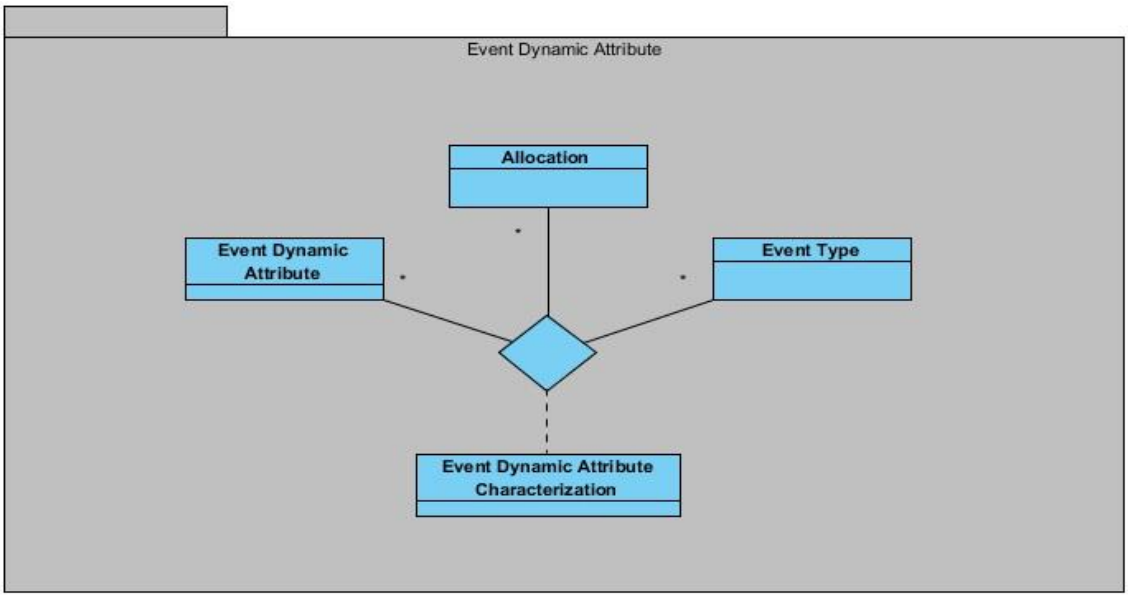


Figura 3-07. Diagrama UML correspondiente a la caracterización de atributos de evento dinámicos

Además, cada atributo de evento puede estar caracterizado por múltiples posibles cambios, es decir, que pueda cambiar de valor si se dan diferentes combinaciones de evento – asignación.

Por último, cabe destacar que los cambios que este tipo de atributos pueden experimentar con la ocurrencia de un evento pueden son:

- Si el atributo es de tipo constante (Constant), de probabilidad (Single Probabilitity), o de distribución (Distribution), el suceso del evento desencadenado por la asignación definida podrá dar lugar a un aumento (Increment) o disminución (Decrement) del valor del atributo en una cantidad determinada por el usuario, o también puede cambiar a un valor concreto definido por el usuario (Change to). Siempre teniendo en cuenta que un atributo de probabilidad no puede valer más de 1 ni menos de 0. En el caso de ser de tipo distribución el parámetro que cambia será la media y la desviación típica si procede.
- Si por el contrario el atributo es de tipo booleano (Boolean), los posibles cambios que sufrirá durante la ejecución del proceso son cambio a un valor determinado (Change to), a verdadero o falso, o un cambio (Change), que si en el momento del cambio el atributo vale verdadero éste cambiará a falso y viceversa.

El proceso de definición de un atributo de evento dinámico ejemplifica muy bien el nivel de abstracción de cada nivel jerárquico presente en el modelo, así como la necesidad de cada uno de ellos en el proceso de definición de propiedades y características hasta llegar al modelo más concreto. En el nivel de instancia se define el tipo de atributo y la entidad de la instancia a la que se asocia, así como los elementos de la instancia a los que se asocia el atributo dinámico. En el nivel de instancia también se lleva a cabo la caracterización del atributo, como se ha explicado anteriormente, definiendo qué eventos producirán un cambio en el valor del atributo, el tipo de cambio que tendrá lugar y qué elemento perteneciente a la entidad "allocation" produce ese evento. En el siguiente nivel, nivel de conjunto de ejecuciones, se define el valor inicial que toma el atributo asociado a cada elemento del modelo. Finalmente, en el último de nivel, nivel de ejecución donde cada elemento es lo más concreto posible, el atributo va

adquiriendo valores concretos en un instante de tiempo concreto y asociado a un elemento de la ejecución concreto, así como se van produciendo cambios de valor en momentos concretos y producidos por eventos concretos.

Por último, cabe destacar que en el nivel de "Conjunto de ejecuciones", tanto para la definición del valor inicial de un atributo de evento dinámico, como para la definición del valor de un atributo dinámico de ejecución constante, serán necesarios los parámetros de atributo que procedan en función del tipo de atributo que se quiera definir, y que han sido ya descritos (Constant value, Logical value, Probability value, etc). Aunque en el caso de atributos de evento dinámicos sólo será necesario un parámetro de atributo para definir el valor inicial de cada atributo; y esto es porque un atributo de evento sólo puede ser contante, de probabilidad única, booleano, o distribución exponencial.

Durante el desarrollo del proyecto, se ha considerado que es necesaria la presencia de una serie de atributos para garantizar la coherencia de los modelos, así como una correcta ejecución/simulación de los procesos definidos. En la figura 3-4 se han incluido alguno de los atributos más significativos, y que son necesarios en el modelo. La función de cada uno se explica a continuación:

- Atributos de la entidad Segmento de Cliente (Customer Segment):
  - *Arrival Distribution Data*: Su utilidad es determinar la distribución de llegadas que siguen los clientes de cada segmento. El acceso de clientes se produce al mismo tiempo que la llegada al sistema. Es un atributo de tipo distribución por definición.

- Atributos de la entidad Conjunto de Infraestructura (Infrastructure Set):
  - *Quantity:* Representa la cantidad de infraestructuras de un tipo concreto para ser asignadas a los clientes que llegan al sistema. Típicamente es un atributo que funciona como variable y cuya optimización se pretende. Su optimización puede ser parte del proceso de dimensionamiento de instalaciones o recursos de una empresa, o tener un carácter estratégico. Un ejemplo es el número de habitaciones simples de hotel, o habitaciones dobles, el número de médicos en un hospital, etc.
  - *Max Waiting List:* Es el número máximo de clientes que se permitirá que estén en la lista de espera para acceder a una infraestructura de un tipo concreto. Se trata de un atributo necesario para el modelo; sin embargo, si el proceso de negocio en definición no contempla lista de espera, el valor adquirido será 0.

- Atributos de la entidad Canal (Channel):
  - *Channel Capacity:* Este atributo reflejará la capacidad que tiene un determinado canal, es decir, el número máximo de clientes que pueden usar el canal. En caso de tener capacidad ilimitada, se deberá asignar un valor muy grande a este atributo, y el canal no supondrá ningún cuello de botella. A modo de ejemplo, puede ser que un hospital cuente sólo con dos ambulancias (canal), o que la web de acceso (canal) sólo permita 10 accesos al mismo tiempo.

- Atributos de la entidad Valor (Value):
  - *Value:* Es el valor que va asociado a una asignación (Allocation) determinada. Este atributo puede representar un valor monetario por la asignación de una infraestructura, como sería el precio, o representar otros posibles valores, como

sería en el caso del sector sanitario. Este atributo también es muy usual que sea utilizado como variable, como el caso de querer optimizar el precio de las habitaciones de un hotel.

- Atributos de la entidad Tiempo (Time):
  - *Time:* Es la unidad de tiempo que se le asignará a cada elemento Time. Como mínimo, siempre debe haber un tiempo inicial (T.Start) y un tiempo final (T.final), pero puede haber tantos hitos temporales intermedios como se quiera.
  - *Clock Value:* Este atributo es fundamental para el proceso de ejecución, ya que hace la función del reloj en dicha simulación. Por lo tanto, es un atributo de tipo constante y dinámico, ya que al ser el reloj irá cambiando de valor con el avance de la ejecución. Se trata de un atributo predefinido y fundamental en el modelo, y marca el tiempo de cada ejecución.

- Atributos de la entidad Acceso a una infraestructura (Infrastructure Access):
  - *Value-Based Decision:* Como se ha explicado en un ejemplo anterior, se trata de un atributo de tipo probabilidad definida a trozos y que recoge la probabilidad de que un cliente que pertenece a un determinado segmento de cliente, y que accede a una infraestructura a traves de un canal concreto, acepte la oferte en función del valor que se ofrece. Al ser a trozos, la probabilidad dependerá del valor asociado.
  - *Service Time:* Es el tiempo de servicio, es decir, el tiempo que el uso de una infraestructura de un determinado tipo permanece reservado por un determinado segmento de cliente que ha accedido por un determinado canal.
  - *Failed Access Probability:* Es la probabilidad de que el acceso de un cliente, una vez que la oferta ha sido aceptada, falle y por consiguiente no se materialice la asignación.

- Atributos de la entidad asignación (Allocation):
  - *Priority:* Es la prioridad con la que una asignación tendrá lugar, es decir, dado un tipo de segmento de cliente que pretende acceder al sistema, se establece un orden de prioridad de asignación, de manera que accederá a través de un canal y a la infraestructura disponible según el orden de prioridad establecido. De esta forma, un determinado cliente accederá siempre a la misma asignación si está disponible. Un ejemplo sería la reserva siempre de la habitación de hotel más barata por parte de un cliente con pocos recursos económicos (Segmento de cliente).
  - *Access Election Probability:* Es la probabilidad de que un cliente que accede al sistema opte por la asignación que ha recibido de acuerdo a la prioridad. El proceso es que primero se opta por una asignación según el nivel de prioridad, y después existe una probabilidad de que el cliente finalmente opte por esa asignación. Puede ser que no siempre interese que el cliente opte por la asignación de acuerdo a la prioridad definida, en ese caso se puede hacer uso de este atributo. No todos los clientes de un mismo tipo optan por la asignación más lógica (dada por la prioridad), y por ello de la existencia de este atributo.
  - *Priority_WL:* Es un concepto similar al de Priority, pero en este caso es para definir el grado de prioridad de aquellos clientes que están en la lista de espera. Establece la prioridad con la que los clientes en lista de espera son asignados a una infraestructura cuando ésta queda disponible. Si se opta por una opción FIFO ("First In, First Out"), en la que los clientes en lista de espera serán asignados a la infraestructura que queda disponible en orden de llegadas, el valor de Priority_WL será de 1 para todas las Allocation.

Además de los atributos explicados, existen dos atributos de evento dinámicos, también necesarios para el modelo y son *"Availability_I"* y *"Availability_C"*. Son de evento dinámico porque su valor cambia durante la ejecución. En el primer caso, sirve para determinar el número de infraestructuras de cada tipo disponibles en cada momento de la ejecución, luego va aumentando y disminuyendo en función de la disponibilidad de infraestructuras. El segundo atributo funciona de manera semejante, y sirve para determinar la disponibilidad de cada canal. Al tratarse de atributos de evento dinámico, han de ser caracterizados por el usuario.

3.4.2 MÓDULO DE DEFINICIÓN DE ESTADOS

Una vez explicado el módulo de atributos, el siguiente módulo necesario en la instancia de proceso, y que la hace más concreta, es el módulo de estados. En este apartado, seguimos dentro del nivel de instancia de proceso, se describe el modelo de estados y el modelo de eventos a través de la clasificación de los mismos. En una primera parte, se describirá el concepto de estado y se explicará el modelo conceptual de estados. En la segunda parte del capítulo, nos centraremos en los tipos de eventos que hay y en la caracterización de los mismos ligado a los atributos definidos.

En este apartado, y para continuar definiendo y concretando el nivel de instancia de proceso de negocio, se definen los estados que caracterizan cada instancia de proceso. Los estados (Status) son las distintas etapas en las que se puede encontrar una entidad asignación (Allocation) durante la ejecución de un proceso de negocio. En todo proceso de ejecución, los clientes que llegan al sistema, a través de un canal para reservar una infraestructura, en un intervalo de tiempo determinado y con un valor asociado, conforman una entidad asignación determinada, que acceden al sistema pasando a través de una serie de estados de acuerdo a unas condiciones que serán definidas por el usuario. Por ejemplo, toda entidad asignación que accede al sistema debe pasar por un estado inicial llamado "Customer Access" (Acceso de cliente); si existe alguna infraestructura disponible para dicho cliente éste pasará al estado "Accepted Access" (Cliente aceptado), y así ira pasando el cliente por el proceso. Dicho cliente pasará por unos estados u otros dependiendo de los valores que adopten los atributos que gobiernan los cambios de estado. La definición de los estados es fundamental tanto para una correcta simulación del proceso, como una buena y más sofisticada forma de definir las funciones objetivos del problema. De esta forma, se añade la opción de incluir en las funciones objetivo términos ligados a la evolución del sistema durante la ejecución del proceso, funcionalidad necesaria para dotar a la herramienta y modelos de una mayor capacidad para la resolución de los problemas en estudio.

En la figura 3-8 se presenta el modelo conceptual perteneciente al módulo de estados, también denominado modelo de la dinámica de una instancia de proceso, y que nos ayuda a explicar y describir el presente módulo. En primer lugar, empezando por la parte superior del modelo, toda instancia de proceso de asignación de infraestructuras deberá tener una definición dinámica de asignación de infraestructuras, que es la denominación que recibe la definición del modelo de estados. En algunos casos se puede presentar instancias de proceso en la que dicha definición no aparezca, pero sería debido a que la definición del nivel de instancia no se ha completado y por ello no se podría pasar a niveles sucesivos (Po este motivo la multiplicidad 0..1).

Toda definición dinámica de asignación de infraestructuras (Modelo de estados) está compuesta por estados (Status), pudiéndose definir tantos como sea necesario. Para que las asignaciones pasen por el sistema de una forma dinámica, éstas irán pasando de un estado a otro en lo que se llama cambio de estado (Status Change). De esta manera, en este módulo se definen los posibles cambios de estado que se pueden experimentar en la ejecución del proceso. Cada cambio de estado esta caracterizado por un estado previo (Previous Status) y un estado siguiente (Next Status), de forma que cualquier estado pude ser el previo y cualquiera el siguiente siempre y cuando el estado previo y el siguiente no coincidan. De esta manera, se definen los diferentes estados presentes en el problema y los posibles cambios de estado que puede experimentar cualquier asignación en su paso por el sistema. Estos estados y cambios de estado se definen de una forma genérica, es decir, de manera que cualquier asignación que acceda al sistema podría pasar por cualquiera de los cambios de estado definidos.

El siguiente paso en el diagrama son los tipos de evento. Un evento es un acontecimiento o suceso que tiene lugar en el sistema durante la ejecución y que puede desencadenar otros acontecimientos. Por lo tanto, la ejecución de todo sistema está gobernada por la ocurrencia de eventos que determinan el funcionamiento y desarrollo de cada ejecución. Como se aprecia en la figura 3-8, los tipos de evento (Event Type) pueden ser o bien eventos de cambio de estado (Status Change Event) o bien eventos de tiempo (Time Event), como muestra la clasificación completa y disjunta.

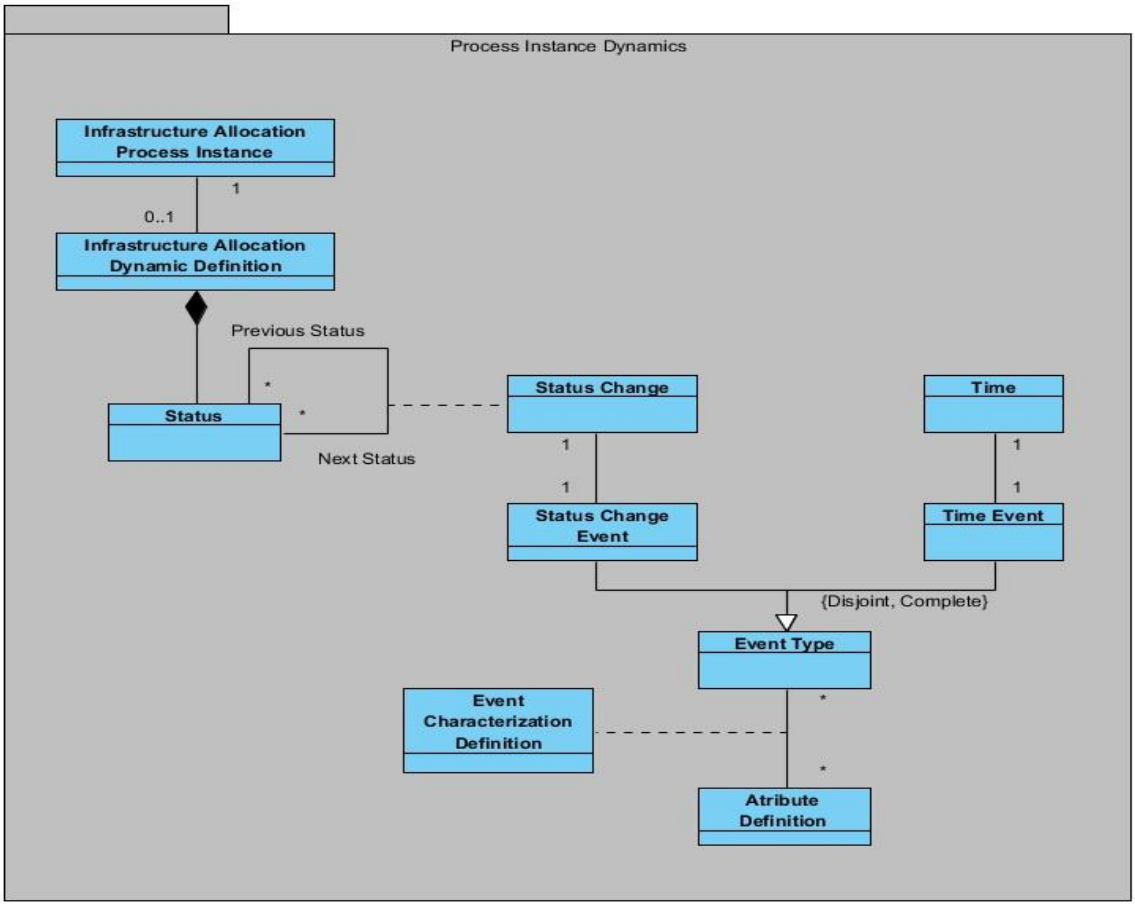


Figura 3-08. Diagrama UML correspondiente al módulo de definición de estados

Asimismo, todo cambio de estado tiene un evento de cambio de estado asociado, como muestra la relación uno a uno del diagrama. Por su parte, todo elemento de tiempo

definido en el modelo tendrá un evento de tiempo asociado, ya que dichos elementos de tiempo suponen hitos en la ejecución. Por ello, los eventos definidos en cada instancia de proceso de negocio estarán compuestos por todos los cambios de estado definidos en este módulo, más los elementos de tiempo definidos en el modelo de proceso y que también son elementos de la instancia.

La parte inferior del diagrama de la figura 3-8 muestra la relación entre los atributos y los eventos. Para que todo evento suceda, ya sea un cambio de estado o un evento de tiempo, se tienen que definir las condiciones que llevan a dicha ocurrencia. En nuestro modelo de estados, los atributos son los encargados de determinar las condiciones necesarias para que suceda cada uno de los eventos ya definidos. Por ejemplo, una asignación pasará del estado "En servicio" al estado "Servido" cuando haya transcurrido el tiempo de servicio, marcado por el valor del atributo "Service Time". En este ejemplo, el cambio de un estado a otro está caracterizado por el atributo "Service Time", y por tanto el evento esta caracterizado por dicho atributo.

Cada tipo de evento puede estar caracterizado por cualquier atributo, de la misma manera que cada atributo puede caracterizar cualquier tipo de evento, siendo posible que un mismo atributo caracterice varios tipos de evento y que un mismo tipo de evento esté caracterizado por varios atributos. Esta relación de muchos a muchos se recoge en la clase denominada definición de caracterización de eventos (Event Characterization Definition). En esta clase se recoge la qué atributo o atributos caracterizan cada uno de los tipos de eventos definidos, así como el valor que ha de tomar el atributo para que el evento suceda. En el capítulo correspondiente al ejemplo de simulación, en la parte final del documento, se expone un ejemplo completo donde se detalla la caracterización de eventos.

En este punto, es importante destacar que han sido identificados una serie de estados y cambios de estado que son necesarios en la definición de toda instancia de proceso. Estos estados y cambios de estado corresponden con la parte inicial del árbol o diagrama de definición estados en la que se define cómo los clientes acceden al sistema. Se trata de estados esenciales para una correcta ejecución de los procesos de acuerdo a los modelos, asegurando de esta manera una óptima resolución de los problemas de asignación de infraestructuras a los que está destinado el proyecto. Tanto la definición de estos eventos como su caracterización mediante atributos, forman parte del modelo conceptual y estarán presentes en toda instancia de proceso, así como en la implementación de la herramienta. Por ser esenciales, no pueden ser modificados. Además, para la caracterización de dichos eventos, se requieren de los atributos esenciales explicados en el apartado anterior, los cuales están también implícitos en el modelo y no se pueden modificar. En la figura 3-9 se presenta el diagrama o árbol de estados de la parte que determina la forma en la que los clientes acceden al sistema y se les asigna una infraestructura. Este diagrama será común a todas las instancias de proceso.

A continuación se explica el diagrama de la figura 3-9 que describe la forma de acceder de los clientes al sistema y la caracterización de los eventos que gobiernan dicho proceso. El proceso comienza con la llegada de un cliente, de acuerdo al atributo Distribución de llegadas (Arrival Distribution), en un momento determinado. Una vez llega el cliente a través de un canal para reservar una infraestructura en un intervalo de tiempo determinado, se le asigna un valor y ese elemento asignación (Allocation) pasa al estado "Customer Access" (Acceso de cliente). Para la simulación de este paso, en el

que se determina por qué canal accede el cliente y a qué infraestructura opta, así como el valor que se le otorga, se requiere de una explicación y un modelo detallado que se explicará en el nivel de ejecución con detalle (apartado 3.6.2). Una vez en dicho estado, la caracterización de eventos que puede experimentar el elemento asignación (Allocation) es como sigue:

- El elemento Allocation puede pasar al estado acceso aceptado y ser aceptado, evento que sucederá cuando la infraestructura asociada a la asignación esté disponible y el atributo Offerable (Ofertable) adoptará un valor True, o pasar al estado acceso rechazado si la infraestructura deseada no está disponible.
- Una vez aceptado el acceso (Accepted Access), la asignación puede pasar a Requested Allocation Access (Petición de acceso), si el valor asociado a la asignación es aceptado por el cliente y que se determina por el atributo Value Based Decision que va ligado al atributo Request, o pasar a ser rechazado si el cliente no acepta el valor asignado.
- Una vez hecha la petición de acceso (Requested Allocation Access), hay una probabilidad de que la asignación de la infraestructura falle por problemas técnicos a la hora del pago o por otro tipo de imprevistos en la formalización de la transacción. La probabilidad de fallo está determinada por el atributo Failed Access, y si no existe fallo la asignación pasará a acceso exitoso (Successful Access).
- Una vez la asignación ha accedido con éxito (Successful Access), ésta pasará a Allocated (Asignado) siempre que la infraestructura ofertada esté disponible, en caso contrario pasará a lista de espera (Waiting List) si el proceso de negocio permite la existencia de lista de espera.
- Si la asignación se encuentra en la lista de espera, ésta pasará a Allocated (Asignado) cuando la infraestructura ofertada a esa asignación quede disponible, y siempre de acuerdo a las reglas de prioridad definidas por el usuario a través del atributo Priority_WL.

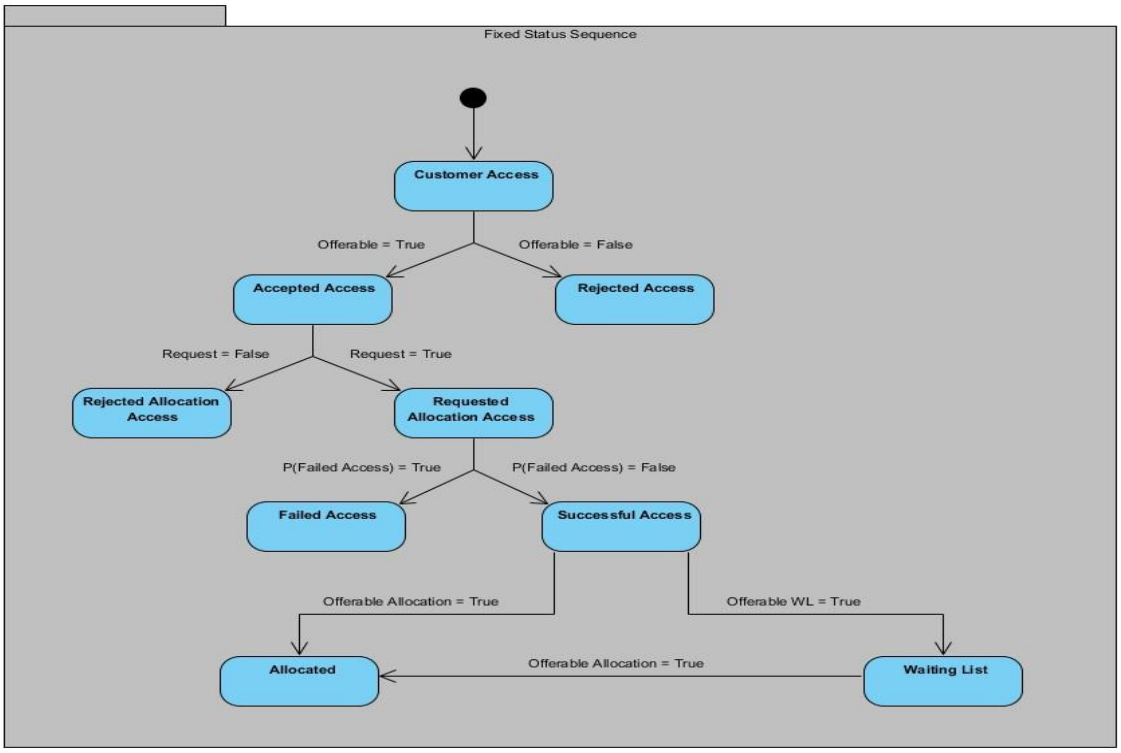


Figura 3-09. Diagrama UML correspondiente al árbol de estados de la parte fija/obligatoria

A modo de ejemplo, un cliente que sigue una tasa de llegadas ajustada a una exponencial de media cinco llega al sistema en determinado momento de la ejecución. Este cliente accede por internet (canal) para reservar una habitación doble (Infraestructura) que en ese intervalo de tiempo (Time) tiene un precio de 100 euros (Value). El cliente llega al estado acceso de cliente, y como la infraestructura deseada está disponible el cliente pasa al estado acceso aceptado. Una vez aquí, el cliente acepta el precio de la oferta y por lo tanto pasará al estado acceso exitoso. Como la habitación ofertada está disponible en ese momento, el cliente pasa al estado asignado (Allocated) y la infraestructura pasa a ser ocupada hasta que el cliente la deje libre. En este punto se abre la posibilidad para el usuario de definir el resto del diagrama de estados de acuerdo a los requerimientos del proceso de negocio bajo estudio.

Volviendo a la definición de tipos de evento, ya hemos explicado que éstos se pueden clasificar en eventos de cambio de estado y eventos de tiempo, pero una clasificación más detallada se muestra en la figura 3-10.

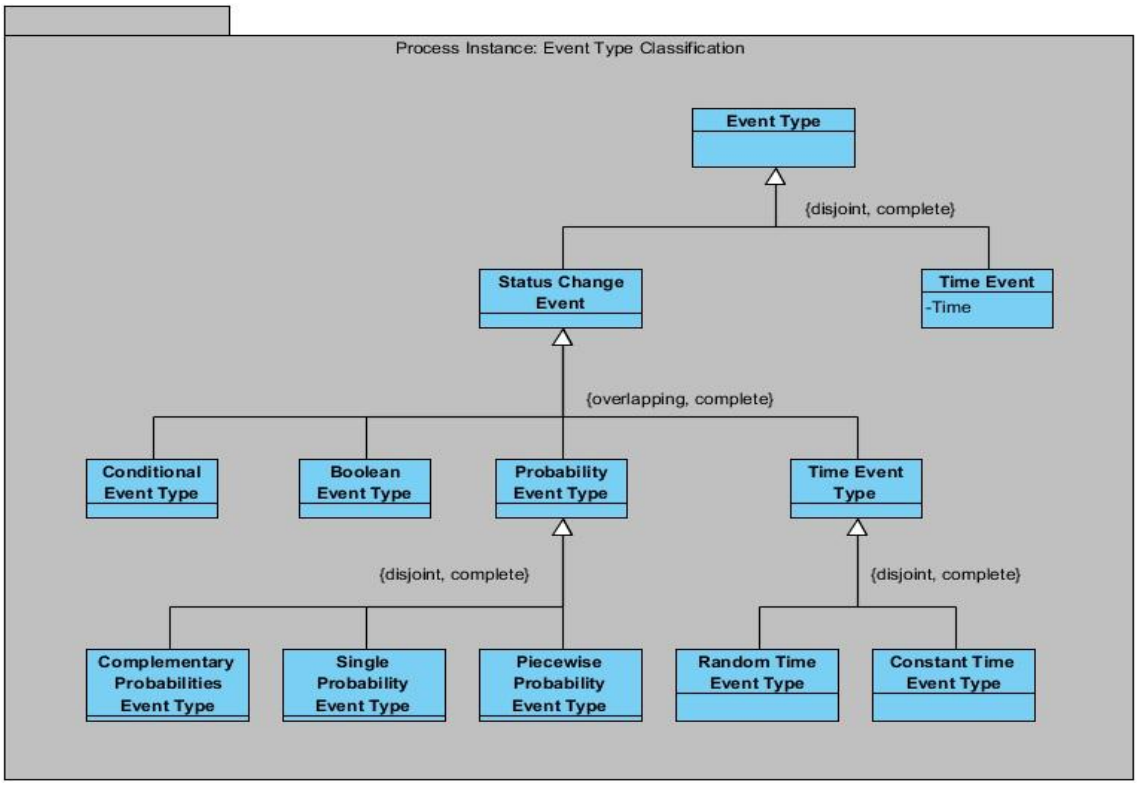


Figura 3-10. Diagrama UML correspondiente a la clasificación de tipos de evento

A la vista de la figura 3-10, la primera clasificación es la mencionada, al ser una clasificación completa y disjunta, un evento debe clasificarse en un evento de cambio de estado o un evento de tiempo. Si es un evento de tiempo supone un hito en la ejecución y estará caracterizado por el valor que adopte dicho tiempo a través del atributo time expresado en el diagrama. Si por el contrario es un evento de cambio de estado, éste se clasificará a su vez de forma en cuatro categorías de forma superpuesta (overlapping) y completa, lo que significa que un tipo de evento de cambio de estado puede pertenecer a más de una de las cuatro categorías que se explican a continuación.

- *Boolean Event Type*: Se trata de tipos de evento de cambio de estado booleanos, que suceden cuando un determinado atributo de tipo booleano toma un valor lógico, verdadero o falso.
- ***Conditional Event Type***: Tipo de evento condicional. Es una de las aportaciones más potentes al proyecto, ya que va ligado a los atributos dinámicos y abre la posibilidad de definir eventos complejos. Se trata de eventos que están caracterizados por atributos dinámicos, cuyo valor como se ha explicado puede cambiar en tiempo de ejecución. La potencia de los atributos dinámicos ligada a la presencia de este tipo de eventos, dota a la herramienta de gran versatilidad y permite ajustar la ejecución de los modelos a la realidad. Este tipo de evento sucederán cuando se cumple la condición definida por el usuario, como podría ser que el número de clientes en lista de espera (Atributo dinámico que exprese el número de clientes en lista de espera en cada momento) sea mayor que cinco, o que el tiempo de ejecución (marcado por el atributo Clock, o reloj de la ejecución) sea superior a 10 minutos. Estos son dos ejemplos de eventos caracterizados por atributos dinámicos que cambian de valor a lo largo de la ejecución.
  Los operadores lógicos que se pueden emplear para definir estas condiciones que deben cumplir los atributos dinámicos para que el evento suceda son: $>$, $\geq$, $\leq$, $<$, o $=$ más el valor con el que comparar. En caso de booleano el operador será =.
- *Time Event Type*: Son tipos de evento de tiempo, pero a diferencia de los eventos de tiempo explicados en la anterior clasificación, éstos son eventos ligados a un atributo que tomará un valor de tiempo determinado, y transcurrido dicho tiempo el evento sucederá. Es decir, no es un evento que afecta a toda la ejecución como supone el hito marcado por un evento de tiempo, sino que es un evento que experimentará una determinada asignación al transcurrir el tiempo marcado por el atributo.

La clasificación anterior es superpuesta (Overlapping) porque un evento de cambio de estado puede estar caracterizado por un atributo o por dos atributos distintos, en cuyo caso el evento sucederá si se cumplen las condiciones marcadas por ambos atributos y definidas por el usuario. En los casos en los que un evento se caracteriza por dos atributos, su clasificación podrá pertenecer a dos categorías, una acorde a cada atributo, y por ello la clasificación anterior es superpuesta.

En una última clasificación, y como se muestra en la figura 3-10, los tipos de evento de tiempo puede clasificarse en última instancia de manera completa en:

- *Tipo de evento de tiempo aleatorio (Random Time Event Type):* Son tipos de evento que suceden tras el paso de un tiempo aleatorio desde que está activo el estado previo del cambio de estado asociado al evento. Un ejemplo sería el tiempo de servicio determinado por el atributo “Service Time” que siga una distribución, de manera que cada asignación pasaría del estado “Allocated” al estado “Served” tras un tiempo aleatorio.
- *Tipo de evento de tiempo constante (Constant Time Event Type):* Igual que en el caso anterior pero aquí el tiempo transcurrido para que suceda el cambio de estado es constante y definido por un atributo.

Siguiendo con la última clasificación, los tipos de evento de probabilidad se clasifican de manera completa en los tres tipos siguientes:

- *Tipos de evento de probabilidades complementarias (Complementary Probabilities Event Types):* Eventos de tipo probabilidad con la característica añadida de que la suma de los valores de probabilidad de los atributos relacionados con el evento debe ser igual a 1.
- *Tipo de evento de probabilidad única (Single Probability Event Type):* Son eventos caracterizados por un valor de una probabilidad del atributo correspondiente. En estos eventos se obtiene un número aleatorio entre 0 y 1, y si dicho número aleatorio es menor que la probabilidad marcada por el atributo el evento alcanza un valor verdadero y tendrá lugar, esto será explicado en detalle apartado de ejecución.
- *Tipo de Evento de Probabilidad definida a trozos (Piecewise Probability Event Type):* Se debe relacionar con otro atributo de tipo constante, que establezca el valor con el que evaluar la probabilidad mediante el atributo de probabilidad a trozos. Una vez evaluada la probabilidad funciona como el caso anterior.

En base a las clasificaciones explicadas, y a la figura 3-9 que muestra los estados necesarios y la caracterización de los eventos necesarios de cambio de estado; en la tabla 3-1 se muestra la clasificación de dichos eventos y los atributos que los caracterizan. Los eventos reciben el nombre del estado al que accede la asignación tras el cambio de estado (Next Status).

Event Type	Classification1	Classification2	Attribute1	Value1	Attribute2	Value2
Customer Access	Random Time	Complementary Probabilities	Arrival Distribution Data		Access Election Probability	True
Accepted Access	Boolean		Offerable	True		
Rejected Access	Boolean		Offerable	False		
Requested Allocation Access	Boolean		Request	True		
Rejected Allocation Access	Boolean		Request	False		
Failed Access	Single Probability		Failed Access Probability	True		
Successful Access	Single Probability		Failed Access Probability	False		
Allocated	Boolean		Offerable Allocation	True		
Waiting List	Boolean		Offerable WL	True		
Waiting List - Allocated	Boolean		Offerable Allocation	True		

Tabla 3-1. Caracterización de eventos de cambio de estado necesarios

En la tabla aparece un ejemplo de un evento caracterizado por dos atributos, y que por lo tanto tiene dos clasificaciones. Se trata del evento acceso de cliente, que sigue una distribución de llegadas, y por ello se clasifica como tiempo aleatorio, y que recibe un tipo de asignación (Elemento de la instancia) dependiendo de una probabilidad complementaria definida por el usuario y marcada por "Access Election Probability" que va ligado al atributo "Priority" ya explicado. Estos eventos han sido explicados con la figura 3-9, y serán detallados nuevamente en el capítulo de simulación.

3.4.3 MÓDULO DE DEFINICIÓN DE LA FUNCIÓN OBJETIVO

Definidos los módulos de atributos y de estados, el tercer y último módulo que termina de definir la instancia de proceso es el módulo de definición de la función objetivo. Este módulo adquiere especial importancia teniendo en cuenta que el proyecto

está dedicado a la optimización de problemas de asignación de infraestructuras de acuerdo a los objetivos o requisitos que marca la función objetivo. En este apartado se explican todos los aspectos importantes en relación a los modelos conceptuales de la definición de la función objetivo, así como la relación con el resto de módulos del modelo conceptual. Para ello, el apartado comenzará con una descripción de los elementos que componen una función objetivo, incluyendo un ejemplo, para pasar a la descripción de modelo conceptual correspondiente a la función objetivo.

Una función objetivo (Objevtive Function) es una expresión matemática que se pretende optimizar, es decir, maximizar o minimizar, o que pretende expresar una restricción que debe cumplir el problema a resolver. La función objetivo es la piedra angular para la optimización y resolución de problemas, por lo que debe expresar tanto las restricciones como los aspectos influyentes del problema para que aporten su peso a la hora de optimizar el problema en base a ellos. La función objetivo es la herramienta empleada para decidir si una solución es mejor que otra en base a los valores de los elementos que componen dicha función.

Los dos anteriores módulos de la instancia, el módulo de atributos y el módulo de estados, eran necesarios para definir cada instancia, pero también son necesarios para poder definir funciones objetivos de forma más flexible y en base a los requerimientos del problema. Por lo tanto, como veremos más adelante, los atributos, estados, y cambios de estado pueden ser empleados en la definición de la función objetivo, ayudando a reflejar y valorar en ella lo sucedido tras cada ejecución. Con esto, la función objetivo podrá tener en cuenta los resultados de cada ejecución.

La función objetivo no sólo supone un módulo necesario para la definición de la instancia de proceso, sino que emplea la funcionalidad aportada tanto por el módulo de atributos como el módulo de estados, ya explicados, para confeccionar la expresión matemática y determinar la solución óptima buscada. La definición de la función objetivo es fundamental, puesto que dependiendo de si el mismo problema posee una u otra función, la solución probablemente variará. Por este motivo, conviene prestar especial atención a la definición de la función objetivo, y plasmar en ella mediante elementos de la expresión, que describiremos a continuación, los aspectos determinantes del problema para que influyan en la solución final. En ella, hay que considerar tanto aspectos positivos como aspectos negativos, y cada elemento con el peso adecuado.

Una función objetivo se puede resolver mediante métodos analíticos, obteniendo una solución exacta, o mediante simulación. Los métodos analíticos son apropiados en sistemas no muy complejos donde se puede obtener una solución relativamente rápida. Sin embargo, a medida que los sistemas se van haciendo más complejos, los métodos analíticos se ven limitados y el problema se vuelve de difícil solución. Por lo tanto, para sistemas más complejos el método de la simulación es más apropiado y obtiene en muchas ocasiones soluciones más cercanas a la realidad, aunque se requiera más recursos para su resolución. En este proyecto se ha escogido la opción de simulación, por considerarse que aporta más flexibilidad, posibilidades, y ajuste a la realidad tanto a la hora de definir problemas complejos como a la hora de solucionarlos. Se trata de un método más potente y fiable para problemas como los estudiados en el presente proyecto que pueden adquirir un alto grado de complejidad.

Como hemos mencionado, una función objetivo es una expresión matemática que se compone de términos o elementos cuya relación mediante operadores matemáticos

conforma la función. De esta manera, podemos ir descomponiendo la función objetivo en términos; en un primer paso tenemos la expresión entera, y en los pasos sucesivos vamos descomponiendo cada dos términos unidos por un operador.

Vamos a apoyarnos en un ejemplo para mostrar el proceso de descomposición en términos. Por ejemplo, tenemos la función objetivo $f = 10 \cdot x + 2 \cdot y$. El primer término es la propia expresión, es decir, $10 \cdot x + 2 \cdot y$. Luego se descompone en dos términos unidos por un operador, y siempre separando por el operador en orden ascendiente de prioridad de operación. Esto quiere decir que si la suma/resta es la última operación que se realiza en una expresión matemática al tomar valores, este operador será el primero en descomponerse, siempre teniendo en cuenta los paréntesis. Por lo tanto, el segundo término sería $10 \cdot x$, y el tercero sería $2 \cdot y$, unidos por el operador +. Siguiendo con la misma descomposición, por un lado tendremos el cuarto término que será 10, y el quinto *x*, unidos por un producto; y por el otro lado tendremos el sexto término que será 2, y el séptimo *y*, vinculados por un producto. En la tabla 3-2 se muestran los diferentes términos identificados en el ejemplo.

ID del Término	Término
1	10·x + 2·y
2	10·x
3	2·y
4	10
5	x
6	2
7	y

Tabla 3-2. Ejemplo de descomposición en términos de una función

Siguiendo con la descripción de la función, cada uno de los términos obtenidos mediante la descomposición se puede clasificar de manera completa y disjunta en términos básicos (Basic Terms) o compuestos (Compound Terms). De esta manera, un término será básico cuando no puede descomponerse en términos más simples, y será compuesto en caso de poder descomponerse. Por lo tanto, todo término compuesto debe estar caracterizado por dos términos más simples y una operación que los vincula. En el ejemplo anterior tenemos cuatro términos básicos (4, 5, 6, y 7) y tres términos compuestos (1, 2, y 3). Cada término compuesto se podrá descomponer en un primer término (First) y en segundo (Second) además de una operación (Operator). La descomposición de términos compuesto se realizará hasta que todos los términos sean básicos y no se puedan descomponer más. En la tabla 3-3 se muestra la composición de los términos compuestos del ejemplo anterior.

ID Operación	Término compuesto	Primer término	Operador	Segundo término
1	10·x + 2·y	10·x	+	2·y
2	10·x	10	·	x
3	2·y	2	·	y

Tabla 3-3. Ejemplo de operaciones en una función

Una vez explicados estos conceptos sobre la función objetivo, ahora nos centraremos en el modelo conceptual desarrollado en el actual proyecto respecto al módulo de definición de la función objetivo perteneciente al nivel de instancia de

proceso. En la figura 3-11 se muestra el diagrama UML del modelo conceptual de la función objetivo. Iremos paso a paso explicando los detalles desde la parte superior del modelo.

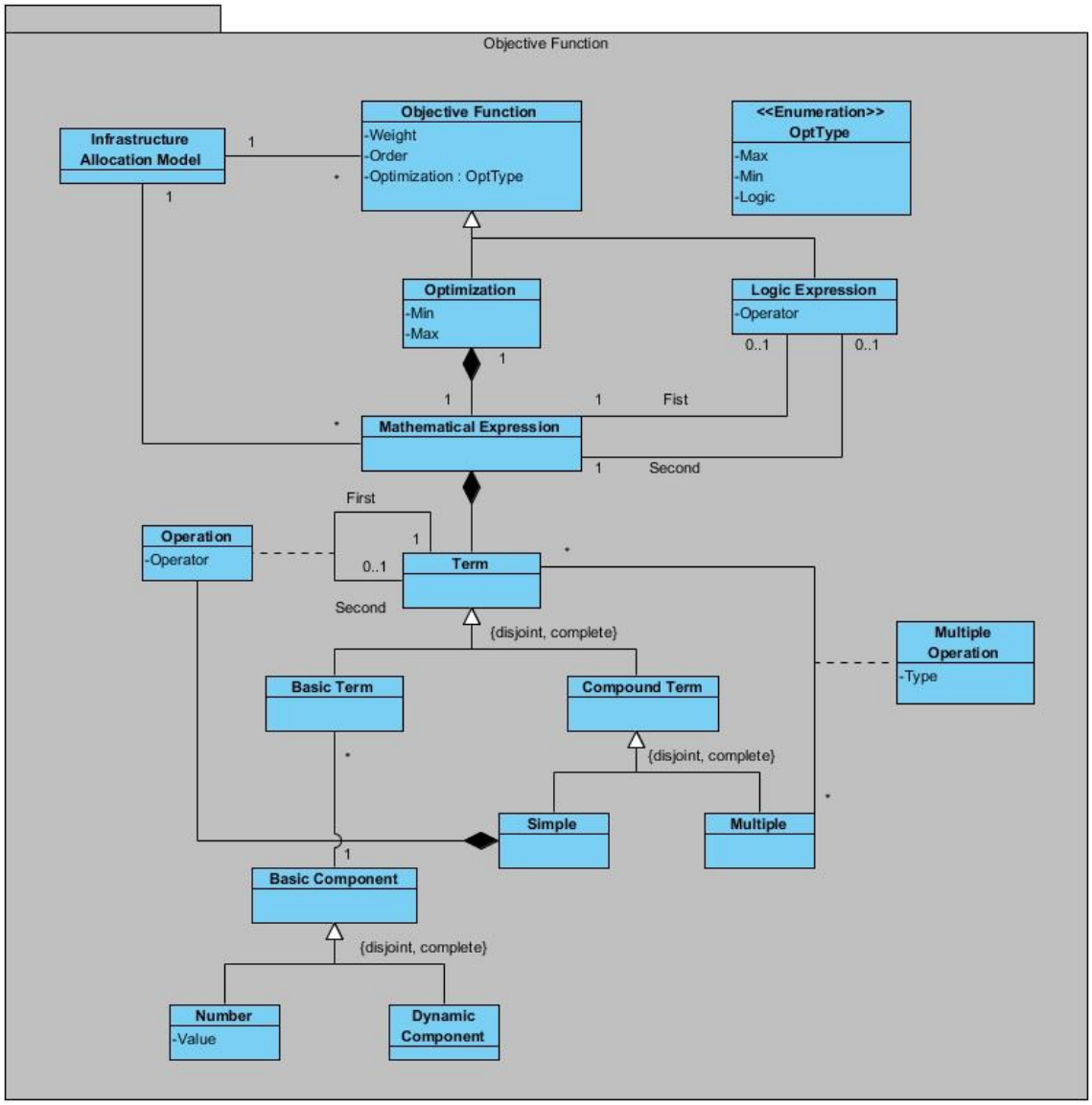


Figura 3-11. Diagrama UML del modelo de función objetivo

Como vemos, un modelo de asignación de infraestructuras (Infrastructure Allocation model) puede tener muchas funciones objetivo. Esto es una de las novedades del proyecto que otorga gran potencial y funcionalidad al modelo. La razón es que todo modelo de asignación de infraestructuras puede requerir para su optimización de la utilización de varias funciones objetivo que reflejen cada una de ellas diferentes aspectos que debe cumplir la solución, como pueden ser restricciones variadas, o la maximización o minimización acorde a diferentes consideraciones que necesite el modelo. Cada función objetivo se caracterizará por tres atributos.

- En primer lugar por el *Orden de Prioridad* entre el conjunto de funciones objetivo definidas, y que se expresa en el diagrama con el atributo *Order*, lo que marcará el orden de prioridad de cumplimiento de dicha función respecto a las restantes. Es decir, si tiene order igual a uno, primero se debe evaluar dicha función y la solución

deberá ajustarse en primera instancia a dicha función, si por el contrario tiene order igual a dos, la solución será de entre las que satisfacen la función objetivo de order 1, la que mejor se ajuste a la función objetivo de order dos.

- En segundo lugar tenemos el atributo *Weight* (Peso). Este atributo representa el grado de influencia de cada función objetivo respecto al total de las mismas. Es decir, es un número ponderado entre cero y uno que marca el peso o importancia de cada función objetivo en el resultado final. Cada función objetivo tendrá un peso, de manera que el resultado numérico resultante de la ejecución de cada función se multiplicará por dicho peso, y el resultado final vendrá dado por la suma de todas las funciones objetivo, multiplicadas por sus correspondientes pesos. La suma de todos los pesos asignados debe ser igual a uno. En caso de no querer emplear este atributo, pues es opcional, se asignará un peso igual a uno a todas las funciones.
- En tercer lugar tenemos el atributo *Optimization Type* (Tipo de optimización). Este atributo es fundamental, y marcará el tipo de función objetivo a definir. En el proyecto se han incluido tres tipos de función objetivo: las funciones de maximización, las de minimización, y las expresiones lógicas que expresarán una restricción del problema. Por lo tanto, este atributo recoge dicha característica, pudiendo tomar tres valores, maximización (Maximization), minimización (Minimization), o lógica (Logic), cada una correspondiente a un tipo de optimización.

Cabe destacar que cada función objetivo será evaluada, y por tanto adquirirá un valor, tras la finalización de cada conjunto de ejecuciones. Esto quiere decir que tras cada conjunto de ejecuciones, cada una de las funciones objetivo tomará un valor de acuerdo al resultado de dichas las ejecuciones. Como explicaremos en el siguiente nivel jerárquico, de cada instancia de proceso pueden instanciarse muchos conjuntos de ejecuciones, cada uno de ellos caracterizados por un valor de las variables y/o atributos dinámicos diferentes. Con esto, los posibles valores que pueden tomar las variables del problema se asignan en el nivel de conjunto de ejecuciones (Execution Set), y de cada conjunto de ejecuciones se instancian muchas ejecuciones (Nivel Execution) para obtener un resultado más realista y preciso.

Siguiendo con el diagrama de la figura 3-11, y como se ha mencionado, toda función objetivo puede ser de dos subtipos, o bien de optimización (Optimization) o bien una expresión lógica (Logical Expression). Si es de tipo optimización podrá ser de tipo **maximización**, donde se pretende que el valor de la función que determina la solución óptima sea el máximo obtenido entre las posibles soluciones, o de tipo **minimización**, en cuyo caso se pretende que el valor que marque la solución óptima sea el menor valor adoptado por la función tras las ejecuciones. Por otro lado, si la función es de tipo **expresión lógica**, ésta representa una restricción del problema, y cuanto más cerca estemos de su cumplimiento mejor será la solución del problema en base a este tipo de función objetivo.

Vemos en el diagrama que cada función objetivo del tipo optimización está compuestas (rombo negro) de una expresión matemática (Mathematical Expression), que habrá que maximizar o minimizar como marca el atributo de este tipo de funciones. Por su parte, toda función objetivo del tipo expresión lógica está definida por una primera expresión matemática (First), una segunda expresión matemática (Second), y un operador definido en el atributo **Operator** (Operador) que determina la relación entre ambas expresiones matemáticas. Este operador, es un operador lógico y podrá tomar los

siguientes valores: =, <, ≤, >, y ≥. En caso de ser definida con el operador igualdad (=), se trata de una ecuación y cuanto más cerca esté de cumplirse mejor será la solución para esta función objetivo. Si por el contrario la función queda definida por alguno de los otros operadores lógicos (<, ≤, >, o ≥), la función es una inecuación y siempre que se cumpla la solución será válida de acuerdo a dicha función.

Además, cada modelo de asignación a infraestructuras puede tener muchas expresiones matemáticas, y no sólo derivadas de las funciones objetivo sino de expresiones matemáticas ligadas a atributos. Aquí se presenta uno de los potenciales puntos a desarrollar en el futuro, y es la vinculación de expresiones matemáticas a atributos, de manera que por ejemplo el valor de un atributo pueda depender del tiempo de ejecución (Reloj de la ejecución). El modelo contempla esta posibilidad de desarrollo futura, que no se ha podido implementar por motivos de alcance del proyecto pero que se ha incluido en el modelado.

Una vez descrita la parte superior del diagrama de la figura 3-11, pasamos a la parte explicada anteriormente en la descripción de una función. Vemos que toda expresión matemática está compuesta de términos (Term). Cada término debe ser básico (Basic Term) o compuesto (Compound Term) en una clasificación completa y disjunta. Como se ha explicado anteriormente, la expresión se descompondrá en términos hasta que todos sean básicos.

Respecto a los **términos compuestos** (Compound Term), cada uno de ellos se puede clasificar de manera completa y disjunta en simple (Simple) o múltiple (Multiple). La clasificación en tipo múltiple (Multiple) fue realizada para contemplar los sumatorios y productorios en la definición de la función objetivo. La descripción de un término múltiple se realiza al final del apartado, tras la explicación de la clasificación de componentes dinámicos. Sin embargo, cabe explicar que cada término compuesto múltiple, por tratarse de un sumatorio o productorio, puede contener varios términos, y que cada término puede estar en más de un término múltiple formando una relación muchos a muchos. Esta relación está caracterizada en la clase Multiple Operation (Operación múltiple), que recoge tanto si es sumatorio o productorio a través del atributo Type (Tipo), como qué términos forman parte de cada término compuesto múltiple, es decir, qué términos pertenecen al sumatorio/productorio definido. Por su parte, un término compuesto simple (Simple) está compuesto por la clase operation (Operación), que consiste en la relación entre dos términos, un primer término (First) y un segundo término (Second), a través de un operador definido por el atributo Operator y en una relación única.

Cada **término básico** (Basic Term), como sería *x* o Id Término 5 del ejemplo de la tabla 3-2, está asociado a un **componente básico** (Basic Component), y distintos términos básicos pueden hacer referencia a un mismo componente básico, como por ejemplo sería el caso si el número 3 (Componente básico) apareciese más de una vez en la función. Los componentes básicos son la piedra angular en la definición de cada función objetivo por parte del usuario, pues representan cada uno de los términos que el usuario define y que se relacionan entre ellos mediante operadores. Por tanto, el usuario define componentes básicos y los relaciona mediante operadores para conformar los distintos tipos de términos explicados hasta la confección de la función objetivo en su totalidad. Todo componente básico se clasifica de manera completa y disjunta en un número (Number), en cuyo caso vendrá definido por un valor y recogido en el atributo Value (Valor), o un **componente dinámico** (Dynamic component), cuya clasificación

se detalla a continuación. En el ejemplo de la tabla 3-2, los términos *x* e *y* serían componentes dinámicos, mientras 10 ó 2 son números.

En la figura 3-12 se muestra el diagrama UML del modelo de la clasificación de un componente dinámico de la función objetivo. Acorde a lo explicado, un componente dinámico es un componente básico, y su valor puede depender del resultado de la ejecución de la instancia de proceso o bien referirse directamente al valor otorgado a los atributos en la fase de definición del problema, siempre que los atributos no sean de evento dinámico pues éstos cambian a lo largo de la ejecución. El matiz fundamental es que los componentes dinámicos no son cifras sin sentido, sino que cada uno tiene consigo implicaciones y carga conceptual por la cual el usuario los ha definido en la función objetivo. Por ejemplo, componentes dinámicos podrían ser el número de médicos cardiólogos en el hospital (Atributo), el número de clientes menores de 50 años atendidos con éxito en un restaurante (número de eventos de cambio de estado), o el tiempo de espera de clientes fieles en un hotel esperando a ser asignados una habitación simple (Tiempo en un estado).

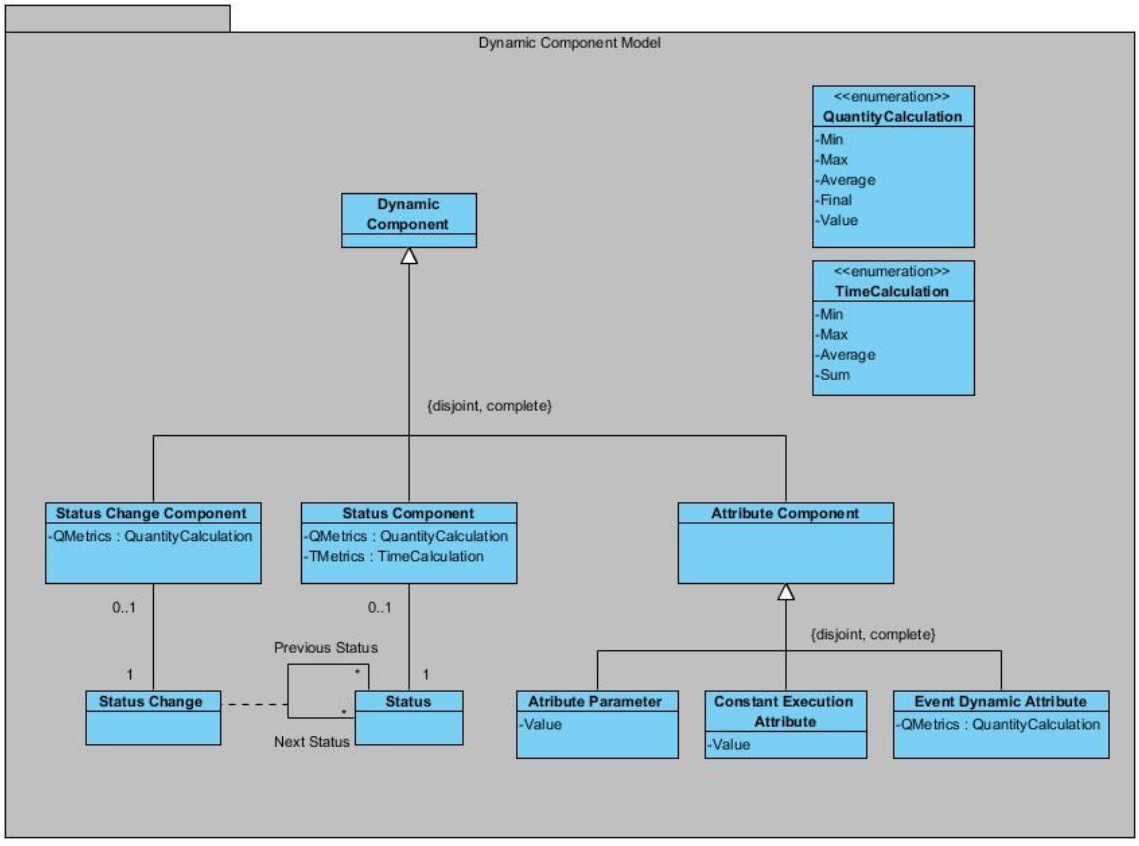


Figura 3-12. Diagrama UML del modelo de componente dinámico de la función objetivo

Los componentes dinámicos se clasifican de manera completa y disjunta en componentes de cambio de estado (Status Change Components), componentes de estado (Status Components), y componentes de atributo (Attribute Components). A continuación se explica cada una de las categorías.

Un componente de cambio de estado (Status Change Component) está asociado a un cambio de estado sucedido a un determinado elemento asignación durante la ejecución de la instancia de proceso. De esta manera, cada vez que se produzca ese

cambio de estado para el elemento asignación definido, este evento contabilizará para función objetivo. Cabe destacar que se trata de un componente que depende del resultado de la ejecución y de la medida de la cantidad elegida, definida en el atributo cálculo de cantidad (Quantity Calculation) y puede ser el mínimo (Min), máximo (Max), número medio (Average) o número final (Final) de cambios de estado para dicho elemento asignación y para todas las ejecuciones realizadas. Es decir, el número medio es el la media de cambios de estado para todas las ejecuciones realizadas para un mismo conjunto de ejecución, puesto que se realizan varias ejecuciones para garantizar resultados más precisos. Un ejemplo sería el número de pacientes graves muertos mientras estaban siendo atendidos en un hospital y por un médico cardiólogo.

Un componente de estado (Status Component), que también depende del resultado de la ejecución, es un componente asociado a un estado concreto y un elemento asignación concreto. En este caso se puede elegir entre una medida de cantidad (Quantity Calculation), que podrían ser el mínimo (Min), máximo (Max), número medio (Average), o número final (Final) de veces que el elemento asignación definido pasa por el estado asociado; o bien una medida de tiempo (Time Calculation), que puede ser el tiempo mínimo (Min), máximo (Max), medio (Average), o total/suma (Sum) que dicho elemento asignación pasa en el estado asociado. Igual que en el caso anterior, los valores mínimo, máximo, media, etc. se calculan para todas las ejecuciones realizadas pertenecientes al mismo conjunto de ejecución (Execution Set). Un ejemplo sería el tiempo medio de espera de clientes fieles en un hotel esperando a ser asignados una habitación simple.

Por último, tenemos los componentes de atributo (Attribute Components). Su valor está asociado a un determinado atributo perteneciente a un determinado elemento del modelo. Como indica la segunda clasificación completa y disjunta, el componente puede estar asociado a un atributo de parámetro estático, a un atributo dinámico de ejecución constante, o a un atributo de evento dinámico. En caso de estar asociado a un atributo estático, ya sea una variable o un parámetro, o a un atributo dinámico de ejecución constante, el valor adoptado por el componente será el valor del atributo (Value) fijado por el usuario antes de las ejecuciones, pues el valor de este tipo de atributos se fija antes de las ejecuciones y es constante a lo largo de las mismas, es decir, no depende del resultado de la simulación. Sin embargo, en caso de ser un componente asociado a un atributo de evento dinámico, el cual puede cambiar a lo largo de las ejecuciones y por ello depende del resultado de las mismas, se podrá elegir medidas de cantidad (Quantity Calculation), tales como el valor mínimo (Min), máximo (Max), medio (Average), final (Final), o inicial (Initial) que alcanza el valor del atributo perteneciente al elemento deseado durante las ejecución de la instancia. Un ejemplo sería el número de médicos cardiólogos en un hospital.

3.5 CONJUNTO DE EJECUCIONES DEL PROCESO

Tras explicar detalladamente los tres primeros niveles jerárquicos del modelo conceptual del proyecto, como son el nivel de metamodelo, el nivel de modelo de proceso de negocio, y el nivel de instancia de proceso de negocio explicado en el apartado anterior; ahora pasamos al cuarto nivel jerárquico llamado nivel de conjunto de ejecuciones del proceso (Process Execution Set) o también llamado "Process Instance Snapshot" (Instantánea de la instancia de proceso). La inclusión de este nuevo nivel respecto al modelo de partida ha sido necesario por motivos que se detallarán a

continuación. El objetivo de este apartado es explicar los aspectos fundamentales del modelo correspondiente a este nivel, así como las implicaciones en el modelo global.

Puesto que el nivel de conjunto de ejecuciones del proceso es el nivel jerárquico siguiente a la instancia de proceso, deberá aportar características nuevas que haga el nivel más concreto respecto al anterior. Como su nombre indica, "Process Instance snapshot" o instantánea de la instancia de proceso, este nivel se puede considerar una foto de la instancia pero añadiendo alguna propiedades y características que hacen que sea considerado una instanciación de la instancia. El desarrollo de los atributos dinámicos a lo largo del proyecto ha sido una de las razones que nos llevaron a la necesidad de este nivel, puesto que como veremos, en este nivel se definen los valores de los atributos dinámicos de ejecución constante y el valor inicial de los atributos de evento dinámicos.

En la figura 3-13 se presenta el diagrama UML del modelo conceptual del nivel de conjunto de ejecuciones del proceso (Process Execution Set). En este nivel sólo es necesario un modelo estático, pues las propiedades añadidas en este nivel corresponden con la asignación de valores a los atributos dinámicos y variables del modelo.

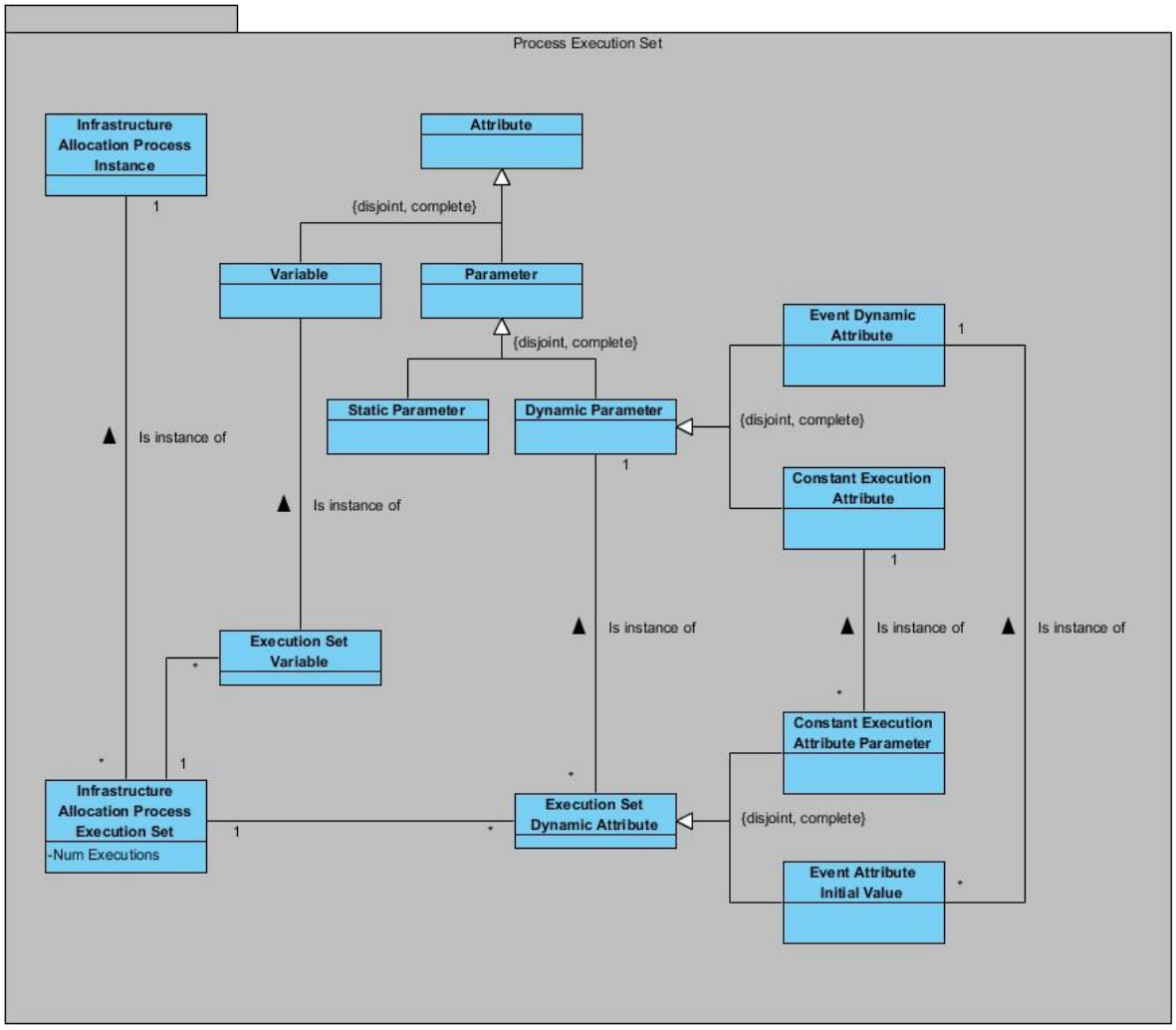


Figura 3-13. Diagrama UML del modelo conceptual del nivel de Process Execution Set

Empezando por la parte superior izquierda del diagrama, vemos que toda instancia de un proceso de asignación de infraestructuras puede tener muchos conjuntos de

ejecuciones del proceso de asignación de infraestructuras (Infrastructure Allocation Process Execution Set), mediante una relación de instanciación del tipo "Is instance of" ya explicada anteriormente. Mediante esta relación, todo conjunto de ejecuciones del proceso que deriva de una instancia, heredará todas las características y métodos de dicha instancia (Padre) y al mismo tiempo aportará más propiedades que convierten al conjunto de ejecuciones en una instancia, con elementos más concretos y definidos. Cada conjunto de ejecuciones del proceso vendrá caracterizado por un número determinado de ejecuciones definido por el usuario a través del atributo Num Executions (Número de ejecuciones), y determina el número de ejecuciones a realizar para en la simulación del proceso para obtener los resultados.

En cada conjunto de ejecuciones del proceso se define el valor que toman las variables del problema y que caracterizan dicho conjunto de ejecuciones respecto a la instancia de la que procede. Esta definición se expresa en el diagrama, donde un conjunto de ejecuciones puede tener muchas variables del conjunto de ejecuciones (Execution Set Variable), que son hijos de las variables definidas en la instancia, pero más concretas al tomar valor en este nivel. La clasificación mostrada en la parte superior del diagrama corresponde con la explicada en el módulo de atributos de la instancia de proceso (Apartado 3.4.1), y reflejada en este nivel pues algunos atributos mediante instanciación adquieren valor.

Por lo tanto, partiendo de la clasificación de atributos, cada conjunto de ejecuciones del proceso puede tener también muchos atributos dinámicos, denominados atributos dinámicos del conjunto de ejecuciones (Execution Set Dynamic Attribute), y que son instanciaciones (Hijos) de los atributos dinámicos definidos en la instancia. En concreto, todos los atributos dinámicos definidos como tal en la instancia adquirirán un valor en este nivel, y dichos valores sirven para caracterizar y diferenciar a los conjuntos de ejecuciones que derivan de una misma instancia. Como vimos en el módulo de atributos, los atributos dinámicos se clasifican de manera completa y disjunta en atributos dinámicos de ejecución constante y atributos de evento dinámicos. Por un lado, los atributos dinámicos de ejecución constante son instanciados para adquirir un valor en este nivel en la clase parámetro de atributo de ejecución constante (Constant Execution Attribute Parameter), y dicho valor será constante a lo largo de las ejecuciones. Por su parte, los atributos de evento dinámicos son heredados por instanciación y se define el valor inicial, a través de la clase valor inicial de atributo de evento (Event Attribute Initial Value), y con el que comenzarán cada ejecución aunque el valor irá cambiando a lo largo ésta.

En resumen, en el nivel de conjunto de ejecuciones del proceso (Process Execution Set) o instantánea de la instancia (Instance Snapshot), tanto las variables del problema, como los atributos dinámicos son heredados por instanciación y cobran valor. Las variables y los atributos dinámicos de ejecución constante tomarán un valor que será constante a lo largo de las ejecuciones, y los atributos de evento dinámico por su parte tomarán un valor inicial con el que comenzar cada ejecución. El resto de propiedades y métodos definidos en la instancia permanecen igual. Por lo tanto, cada conjunto de ejecuciones se caracterizará y diferenciará del resto de conjunto de ejecuciones que pertenecen a la misma instancia en tres aspectos: El número de ejecuciones necesarias, el valor de las variables del problema, y el valor de los atributos dinámicos.

Tras la explicación del modelo, cabe destacar que en este nivel no es necesario instanciar los elementos de la instancia (Instance Element), ya que conceptualmente

habrá diversas "versiones" (Snapshots) de cada elemento según los valores que tomen sus atributos. Por ejemplo, un conjunto de ejecuciones puede caracterizarse por clientes menores de 25 años con una distribución de llegadas (atributo Arrival Distribution) que sigue una exponencial de media 10, y otro conjunto de ejecuciones perteneciente a la misma instancia caracterizada por clientes menores de 25 años con distribución de llegadas exponencial de media 15. Se trata del mismo elemento de la instancia en diferentes "versiones" o instantáneas (Snapshots). Por este motivo, el nivel de conjunto de ejecuciones se puede considerar una "foto" o "instantánea" (Snapshot) de la instancia, donde cada conjunto de ejecuciones es una "instantánea" distinta, diferenciada en los aspectos explicados. Esta consideración es debida a que en este nivel no se aportan más características que los valores explicados, el resto permanece con el mismo grado de definición, y por ello se puede considerar que hay diversas versiones en función de estos valores.

Para concluir la explicación de este nivel, y retomando el ejemplo que se planteo al inicio del capítulo, en la figura 3-14 se presenta un pequeño ejemplo que puede resultar ilustrador.

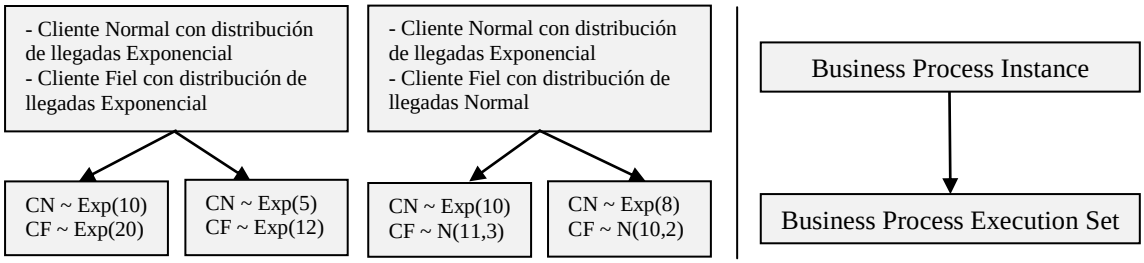


Figura 3-14. Ejemplo de diferentes conjuntos de ejecuciones

3.6 EJECUCIÓN DEL PROCESO DE NEGOCIO

Entramos en el quinto y último nivel jerárquico del modelo, el nivel de ejecución del proceso de negocio (Business Process Execution), donde todos los elementos son lo más concreto posible. Una vez que el usuario ha definido perfectamente los parámetros del problema y los ha ajustado al entorno específico del proceso de negocio, pasamos a resolver el problema mediante simulación, que es la parte correspondiente a este nivel. Al emplear la simulación como medio de resolución de los problemas de asignación de infraestructuras en estudio, primero conviene explicar aspectos importantes de este método.

Por lo tanto, el objetivo de este apartado será en primer lugar describir los elementos y características más importantes de una simulación, y posteriormente describir detalladamente los modelos conceptuales que fundamentan este nivel jerárquico y que han sido desarrollados durante el proyecto. Con esto, el apartado se dividirá en dos partes. En una primera parte se define qué es la simulación, la clasificación de los modelos de simulación, se discutirán los métodos de avance del tiempo en la simulación, así como la elección de la simulación frente a otros métodos y la generación de números aleatorios. En la segunda parte del apartado se detallarán los modelos conceptuales de este nivel y que fundamentan la simulación de los procesos de negocio en estudio.

3.6.1 SIMULACIÓN

Una simulación es el proceso por el que se imita el comportamiento de un sistema real durante un espacio temporal definido y conocido como tiempo de simulación. Los sistemas reales son por lo general muy complejos, por lo que se recurre a la asunción de hipótesis para modelar dichos sistemas y son estos modelos (modelos de simulación) los que realmente se simulan. Por ello, cuanto más parecidos sean los modelos a la realidad más precisos serán los resultados de la simulación. En nuestro caso, uno de los objetivos del proyecto es lograr desarrollar unos modelos conceptuales que permitan ajustarse lo más posible a la realidad de los procesos de negocio en estudio, para lograr mediante simulación resultados suficientemente precisos como para servir a la toma de decisiones (DSS).

En aquellos sistemas en los que la complejidad de los mismos es un problema para ser resueltos por métodos analíticos, la simulación cobra especial importancia. Nos apoyamos en la capacidad computacional de los ordenadores para simular los modelos de simulación. Según afirman A.M. Law y W.D. Kelton en su obra *Simulation Modeling and Analysis* (7), los modelos de simulación se pueden clasificar respecto a tres dimensiones distintas:

- *Estático vs Dinámico*: Un modelo de simulación estático es aquel en el que el tiempo no juega ningún rol, es decir, es la representación de un sistema en un determinado momento del tiempo. Sin embargo, un modelo de simulación dinámico representa a un sistema que evoluciona con el paso del tiempo.
- *Determinista vs Estocástico*: El modelo es determinista si no contiene ningún componente aleatorio. En este caso de tener algún componente aleatorio, como una distribución, el modelo será estocástico.
- *Continuo vs Discreto*: Dependiendo de si las variables de estado del modelo de simulación cambian instantáneamente en puntos distantes de tiempo (Discreto) o lo hacen de manera continua (Continuo).

En nuestro caso, de acuerdo a las clasificaciones anteriores, el modelo de simulación elegido es dinámico, estocástico, y discreto. Es dinámico porque evoluciona con el paso del tiempo, estocástico porque contiene componentes aleatorios como la llegada de clientes al sistema (Distribución) o los componentes de probabilidad, y discreto porque las variables de estado sólo pueden cambiar en un número finito de puntos temporales marcados por la ocurrencia de eventos. Es una simulación discreta basada en eventos, siendo un evento un suceso instantáneo que puede modificar el estado del sistema.

3.6.1.1 MECANISMOS DE AVANCE DEL TIEMPO

En los modelos de simulación dinámicos, como es nuestro caso, existen dos alternativas para tratar el avance del tiempo en la simulación:

- *Avance del tiempo hasta el siguiente evento*: Es el enfoque usado por la mayoría de software de simulación y el elegido para el presente proyecto. Empezando por el tiempo de comienzo de la simulación, se determina el tiempo de ocurrencia de los sucesivos eventos y se avanza el reloj hasta el evento más temprano, el cual actualiza el valor de las variables de estado. Este mecanismo de avance del reloj de

la simulación de un evento al siguiente más temprano se realiza hasta la condición de fin de simulación o el tiempo de fin de simulación.

- *Avance del tiempo a intervalos fijos*: El reloj de simulación se incrementa en unidades o pasos de tiempo constantes, Δt. Después de cada incremento, se comprueba si en ese lapso temporal debería haber sucedido algún evento. En caso afirmativo, se trata a dichos eventos como si hubiesen ocurrido al final del Δ*t*. Este mecanismo del avance del tiempo nos obliga además a definir una regla de preferencias de ejecución para aquellos casos en los que dos o más eventos coincidan en el mismo paso temporal.

La elección para el modelo de simulación de nuestro proyecta ha sido un avance del tiempo hasta el siguiente evento. Este método tiene dos ventajas importantes respecto al otro método explicado. Por un lado, el reloj sólo avanza hasta el tiempo en el que suceden eventos evitando los tiempos de inactividad, lo que requiere mucho menos tiempo de computación que en el caso de un avance a intervalos fijos. Por otro lado, cada evento sucede en el tiempo exacto que le corresponde, y no al final de un intervalo Δt que hace perder precisión al resultado de la simulación. Se trata pues de un método más eficiente y con resultados suficientemente precisos.

3.6.1.2 COMPONENTES FUNDAMENTALES DE UN MODELO DE SIMULACIÓN DISCRETA BASADA EN EVENTOS

En un modelo de simulación discreta basada en eventos y con avance del tiempo hasta el siguiente evento, los siguientes elementos son fundamentales:

- *Reloj de la simulación*: Marca el tiempo de la simulación, es dado por el atributo clock Value (De la entidad Time).
- *Lista de Eventos*: Enumeración de los eventos que se sabe que van a suceder, ordenados por orden cronológico. Es una lista dinámica pues cada evento puede hacer suceder o cambiar otros eventos.
- *Rutina de inicialización*: Inicializa el tiempo al comienzo de la simulación.
- *Rutina del tiempo*: Subprograma que determina el siguiente evento de la lista, y avanza el reloj de simulación hasta el tiempo en que dicho evento está programado.
- *Rutinas de eventos*: Subprograma que actualiza el estado del sistema cuando un particular tipo de evento ocurre. Puede modificar la lista de eventos. Cabe decir que existe una rutina de eventos distinta para cada tipo de evento.
- *Generador de números aleatorios*: Provee a las rutinas de eventos de números aleatorios. Será explicado en el apartado 3.6.1.4.
- *Programa principal*: Es el "Main", el cual gobierna la simulación.
- *Generador de estadísticas*: Subprograma que realiza los cálculos necesarios al final de la simulación para la presentación de resultados al usuario o su almacenamiento. Estará ligado a la evaluación de las funciones objetivo definidas.

3.6.1.3 SIMULACIÓN FRENTE A ALGORITMO

A la hora de elegir la simulación como medio de resolución de los problemas de Revenue Management en estudio, se han tenido en cuenta varias características que este método presenta respecto a una resolución por métodos analíticos:

- Puesto que los sistemas en estudio están asociados a fenómenos estocásticos, no existe una única solución analítica partiendo de un determinado estado inicial.
- La simulación permite anticipar el comportamiento del sistema y probar diferentes alternativas de actuación sin la necesidad de experimentar sobre el sistema real, lo cual podría ser muy costoso.
- Una vez programada la herramienta y definido el problema, la simulación hace posible repetir un experimento tantas veces como sea necesario con un coste adicional despreciable.

Además, hemos tenido en cuenta un aspecto fundamental que diferencia ambos métodos. Mediante simulación se pueden afrontar un amplio rango de problemas con un mismo modelo de simulación, aspecto fundamental cuando el objetivo del proyecto ha sido desarrollar unos modelos teóricos genéricos válidos para un amplio abanico de problemas de asignación de infraestructuras. Por su parte, en los métodos analíticos, dependiendo de qué problema se quiera abordar un algoritmo funcionará mejor que otros, y esto obligaría a tener una amplia librería de algoritmos diferentes que habría que desarrollar uno a uno y seleccionar el más apropiado para cada problema. Además, aunque se escoja el más apropiado, no sería sencillo ajustar los parámetros del algoritmo a los datos de entrada del problema.

3.6.1.4 GENERACIÓN DE NÚMEROS ALEATORIOS

En todo proceso de simulación de cualquier sistema o proceso en el que hay componentes aleatorios (Estocásticas), se requiere la utilización de métodos que generen número que puedan considerarse aleatorios. En el contexto del proyecto que nos ocupa, la generación de números aleatorios será esencial para la obtención de variables aleatorias que puedan garantizar una mejor aproximación de los datos a la realidad.

En la simulación del modelo desarrollado, los números aleatorios generados son necesarios para simular, de forma eficiente y realista, los atributos de tipo tanto distribución (Distribution) como probabilidad (Probability). Gracias a dichos números aleatorios, podemos obtener inputs para la simulación del modelo de manera que se garantice la independencia de los datos de entrada.

Generaremos variables aleatorias a partir de una distribución uniforme en el intervalo [0, 1], denominada U(0, 1). Por lo tanto, las variables aleatorias generadas a partir de la distribución U(0, 1) se llamarán *números aleatorios.* Existen diversos y muy complejos métodos para generar números aleatorios, pero nuestro propósito es hacerlo de manera eficaz, sin excesiva complejidad, pero sobre todo de manera que garantice la independencia de los mismos. Para nuestro proceso de simulación, vamos a emplear un generador de números aleatorios que es ampliamente utilizado en entornos de simulación. Se trata de un generador de números aleatorios de tipo *"Generador de congruencia lineal"* (***linear congruential generator LCG***), introducido por Lehmer (1951). Una secuencia de enteros es definida por la fórmula recursiva:

$$\boldsymbol{Z_i = (aZ_{i-1} + c)(mod\ m)} \qquad (3.1)$$

m: Módulo
a: Multiplicador
c: Incremento

Z_0: *Semilla o Valor inicial (Mayor que cero)*

Según la expresión (3.1), para obtener un número aleatorio de la serie (Z_i), debemos dividir la expresión $\boldsymbol{aZ_{i-1} + c}$ entre **m**, donde Z_{i-1} es el número aleatorio de la serie previo al número que buscamos. Tras dicha división, el resto que obtenemos será asignado al número buscado Z_i. Al tomar números que equivalen al resto de una división, tenemos la condición necesaria $0 \leq Z_i \leq m - 1$; y para obtener el número aleatorio deseado U_i (para i=1, 2, …) en el intervalo [0, 1], tomaremos $\boxed{U_i = Z_i/m}$.

A parte de la condición de no-negatividad de la semilla, se deben satisfacer las siguientes reglas: m > 0, m > a, m > c, m > Z_0. Además, una correcta elección de los parámetros es esencial para garantizar que los correspondientes U_i parezcan ser IID (Independent and Identically Distributed)

Por último, puede ser de gran uso la reproducibilidad que nos permite este método, con el fin de poder comparar simulaciones de distintos sistemas o procesos. Si tomamos siempre los mismos valores de los parámetros m, a, y c, sólo tenemos que recordar la semilla Z_0 para generar los mismos valores para U_i´s. De la misma manera, podemos fácilmente reanudar la generación de Z_i´s en cualquier punto de la secuencia guardando la Z_i final obtenida previamente y usándola como la nueva semilla.

En el caso de nuestra simulación, para evitar los problemas que puedan surgir derivados de este método, y satisfacer los requisitos que garanticen una generación de números aleatorios correcta, se emplearán los siguientes valores para los parámetros necesarios:

$$\boldsymbol{m} = 2^{31} - 1$$
$$\boldsymbol{a} = 7^5 = 16807$$
$$\boldsymbol{c} = 0$$

Con lo que la expresión (3.1) quedará de la siguiente manera:

$$\boldsymbol{Z_i = (16807 \cdot Z_{i-1})(mod\ (2^{31} - 1))}$$

Transformada inversa

En nuestro modelo de atributos, algunos serán definidos como tipo "distribution", y se ajustarán a una distribución para tomar valores a lo largo de la ejecución. Para estos atributos, no es suficiente con obtener números aleatorios en [0, 1] y haremos uso de la transformada inversa para obtener el valor que adquirirá la distribución en cada instante.

Una vez tenemos las IDD variables aleatorias que se ajustan a una U(0, 1), tenemos que generar las variables aleatorias que se ajustan a una función de distribución continua F, teniendo en cuenta que $0 < F(x) < 1$. Denotamos F^{-1} como la inversa de la función de distribución F. Entonces, el proceso para generar una variable aleatoria que se ajuste a la función de distribución F es el siguiente:

1. Generar U~U(0, 1) (como hemos visto anteriormente)
2. Obtener $X = F^{-1}(U)$

Señalar que $F^{-1}(U)$ siempre estará definida, puesto que $0 \leq U \leq 1$ y el rango de F es [0, 1]. En la figura x.1 se muestra cómo el número aleatorio U resulta en la variable aleatoria X, empleando el *método de la transformada inversa.*

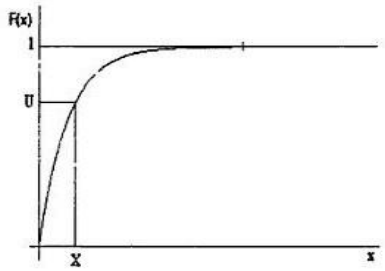


Figura 3-15. Método de la transformada inversa para variables aleatorias

Por ejemplo, la distribución exponencial, muy usada para simular la llegada de clientes a un sistema y por lo tanto propicia para nuestro proceso de simulación, tiene las siguientes características:

$$F(x) = \begin{cases} 1 - e^{-x/\beta} & si\ x \geq 0 \\ 0 & si\ x < 0 \end{cases}$$

Si fijamos u=F(x), entonces: $F^{-1}(u) = -\beta ln(1-u)$

Por lo tanto, primero generamos U~U(0, 1) y después obtenemos:

$$\boldsymbol{X = -\beta \ln (U)} \qquad (3.2)$$

Es posible en este caso reemplazar U por 1-U, puesto que U y 1-U tienen la misma distribución U(0, 1). Con esto, U lo obtendremos con el generador de números aleatorios explicado anteriormente, y β es el parámetro definido por el usuario.

3.6.2 MODELOS CONCEPTUALES DEL NIVEL DE EJECUCIÓN

En este último apartado del modelo teórico, profundizaremos en los modelos del último nivel jerárquico, el nivel de Ejecución del proceso (Process Execution). Este nivel es lo más concreto posible, donde se simula el comportamiento del sistema en estudio, y se realiza mediante la simulación de la llegada de clientes concretos, en un tiempo concreto, y con unos parámetros concretos, que van pasando por estados y sufriendo cambios concretos. Una vez se han explicado todos los niveles superiores, y los conceptos de simulación, en este apartado se describen los modelos conceptuales desarrollados y los aspectos más destacados. Para ello, explicaremos en primer lugar el modelo estático, y posteriormente la dinámica del proceso de ejecución.

En este nivel, nivel de ejecución del proceso (Process Execution), es donde tiene lugar la simulación del funcionamiento de los procesos de negocio bajo estudio. Una vez se han definido todos los parámetros necesarios a lo largo de los cuatro niveles anteriores, en este nivel sólo falta llevar a cabo la simulación basada en el modelo de simulación elegido y en los modelos conceptuales explicados a continuación.

En primer lugar, y como se muestra en la figura 3-16, surge la necesidad de definir un modelo estático del nivel de ejecución, en el que los elementos del modelo de asignación de infraestructuras son heredados en el nivel de ejecución. El diagrama UML de la figura 3-16 es muy similar al explicado en el apartado 3.3 (Figura 3-4), en el que de cada entidad del modelo podían instanciar o derivar muchos elementos del modelo. En este caso, partiendo de la esquina inferior derecha, cada conjunto de ejecuciones del proceso tendrá una o muchas ejecuciones en una relación de instanciación, ya que el nivel de ejecución es más concreto. Por su parte, cada ejecución del proceso (Process

Execution) tendrá muchos objetos de ejecución (Infrastructure Allocation Execution Object), cada uno de los cuales será una instancia del correspondiente elemento del modelo (Model Element). Es decir, cada objeto de ejecución pertenecerá necesariamente, debido a la clasificación completa y disjunta, a una de estas clases: cliente de ejecución, canal de ejecución, tiempo de ejecución, infraestructura de ejecución, infraestructura de acceso de ejecución, valor de ejecución, o asignación de ejecución. Cada uno de estos objetos son instanciaciones de los elementos correspondientes, pues son más concretos al estar totalmente definidos, es decir, tienen todas las propiedades totalmente definidas y forman parte de la simulación en un momento concreto y sufren cambios concretos.

De esta manera, mediante relaciones de instanciación, cada elemento definido en la instancia podrá tener muchos objetos en el proceso de ejecución, objetos que heredan las mismas propiedades pero al ser parte del proceso de simulación del sistema están totalmente caracterizados. Por ejemplo, el elemento del modelo habitación de doble de un hotel (Infraestructura), que en la instancia o en el nivel de conjunto de ejecuciones estaba caracterizado por tener una cantidad de 20 (20 habitaciones dobles), en el nivel de ejecución cuando un cliente determinado accede a una habitación doble concreta (de entre las 20 disponibles) y en un tiempo concreto, esta habitación es un objeto de ejecución y es una instancia del elemento habitación doble definido.

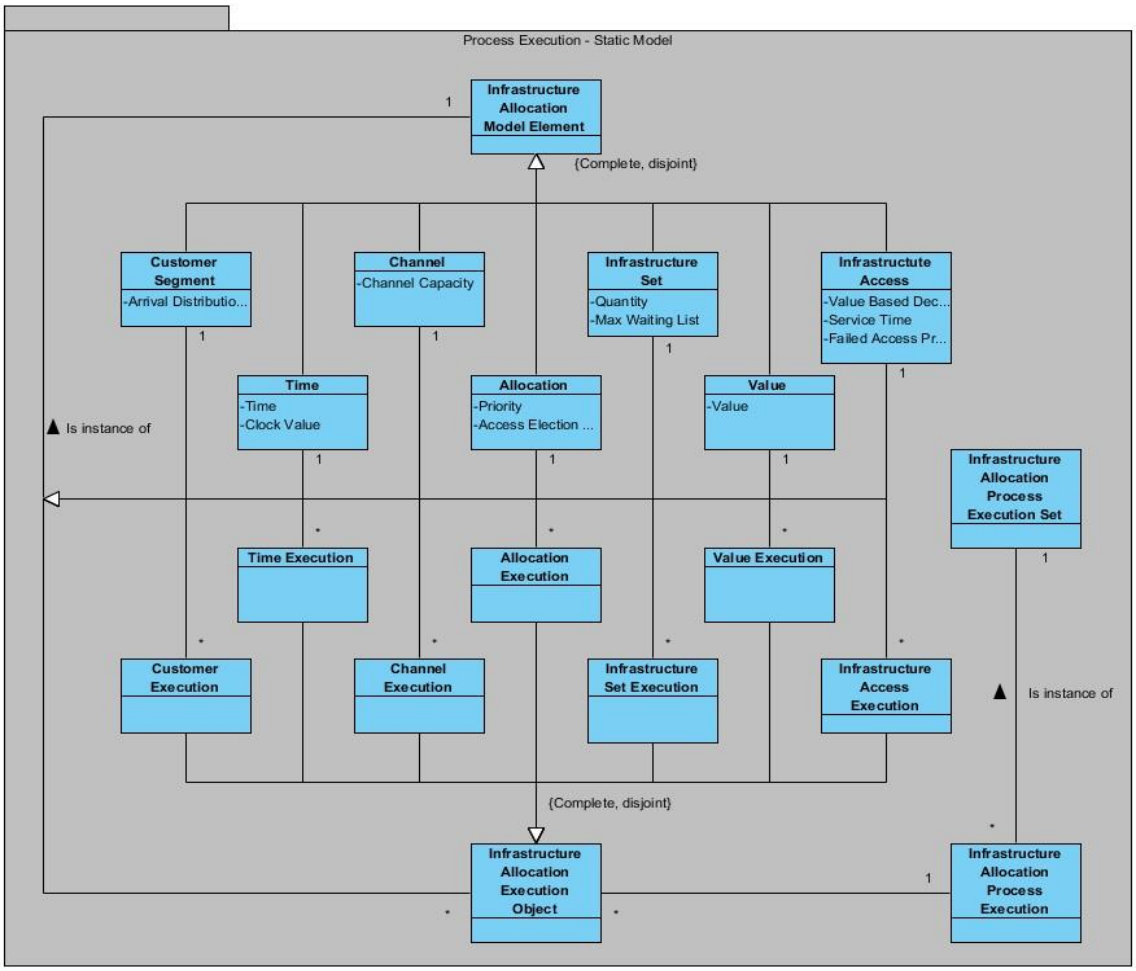


Figura 3-16. Diagrama UML del modelo estático del nivel de Process Execution

Finalmente, pasamos al último diagrama dentro del modelo conceptual del proyecto. En la figura 3-17 se muestra el diagrama UML perteneciente a la dinámica del

proceso de ejecución/simulación del modelo de negocio definido. Dicho diagrama define cómo tiene lugar la simulación y la dinámica de funcionamiento.

En primer lugar, es necesario explicar la llegada de los clientes al sistema. Cada elemento cliente del modelo llegará al sistema de acuerdo a la distribución de llegadas marcada por el atributo "Arrival Distribution". La clase del diagrama Infrastructure Execution Status (Estado de ejecución de infraestructuras), a través del atributo Availability _I (Disponibilidad de infraestructuras) contiene la disponibilidad de cada elemento infraestructura. Por su parte, la clase Offerable Infrastructure (Infraestructura ofertada) es un subtipo o subconjunto de la anterior clase, y contiene sólo las infraestructuras disponibles en cada momento. El mismo procedimiento ocurre para los elementos Canal (Channel), donde la clase Channel Capacity (Capacidad del canal), subtipo de Channel Execution Status, recoge los elementos canal que están disponibles en cada momento. Esto es porque tanto las infraestructuras como los canales pueden tener capacidad limitada. De entre las infraestructuras y los canales disponibles en cada momento, cada elemento cliente tendrá un/os accesos a infraestructuras (terna cliente, canal, infraestructura) ofertadas en dicho momento, y que serán instancias del elemento Infrastructure Access correspondiente. Los accesos a infraestructuras ofertados en cada momento, junto con los valores posibles para dichos elementos, conforman la clase Offerable Allocation, asignaciones ofertadas, que son instancias del elemento asignación definido en la instancia. Por lo tanto, cada vez que llega un cliente de ejecución concreto (Customer Execution) al sistema, se consultan las asignaciones disponibles en ese momento para ese tipo de cliente y forman la clase Consulting (Consultoría), en la que figurarán todos los objetos asignación que se pueden dar en ese preciso momento.

Una vez que tenemos la clase consulting con todos los posibles objetos asignaciones disponibles en ese momento, se determina qué asignación es la que tiene lugar a través de los atributos Priority (Prioridad) y Access Election Probability (Probabilidad de elección de acceso), ya explicados en el módulo de atributos de la instancia. Primero se elige el objeto asignación de acuerdo al orden de prioridad marcado por el atributo Priority, y posteriormente se evalúa la probabilidad del objeto elegido. Si la probabilidad es cierta, entonces esa asignación tendrá lugar, en caso contrario se evalúa la probabilidad del siguiente objeto asignación de la lista de prioridades, y así sucesivamente. Por tanto, el objeto asignación elegido en cada momento es recogido en la clase Execution Infrastructure Allocation (asignación de infraestructura de ejecución), y será un subtipo de la clase consulting.

Este objeto asignación elegido, pasará por el sistema a través de los diferentes estados definidos, y su paso por los estados genera la clase estado de ejecución de asignación de infraestructuras (Infrastructure Allocation Execution Status). Para pasar de un estado a otro se necesita un cambio de estado (Cambio de estado de ejecución), que a su vez es un evento, como se explicó en el módulo de estados.

Siguiendo nuestro modelo de simulación, éste está gobernado por una lista de eventos, que recoge tanto los eventos pasados como los programados en el futuro y que marcan el tiempo y estado de la simulación. Esta lista de eventos es la lista de eventos de ejecución (Execution Event List).

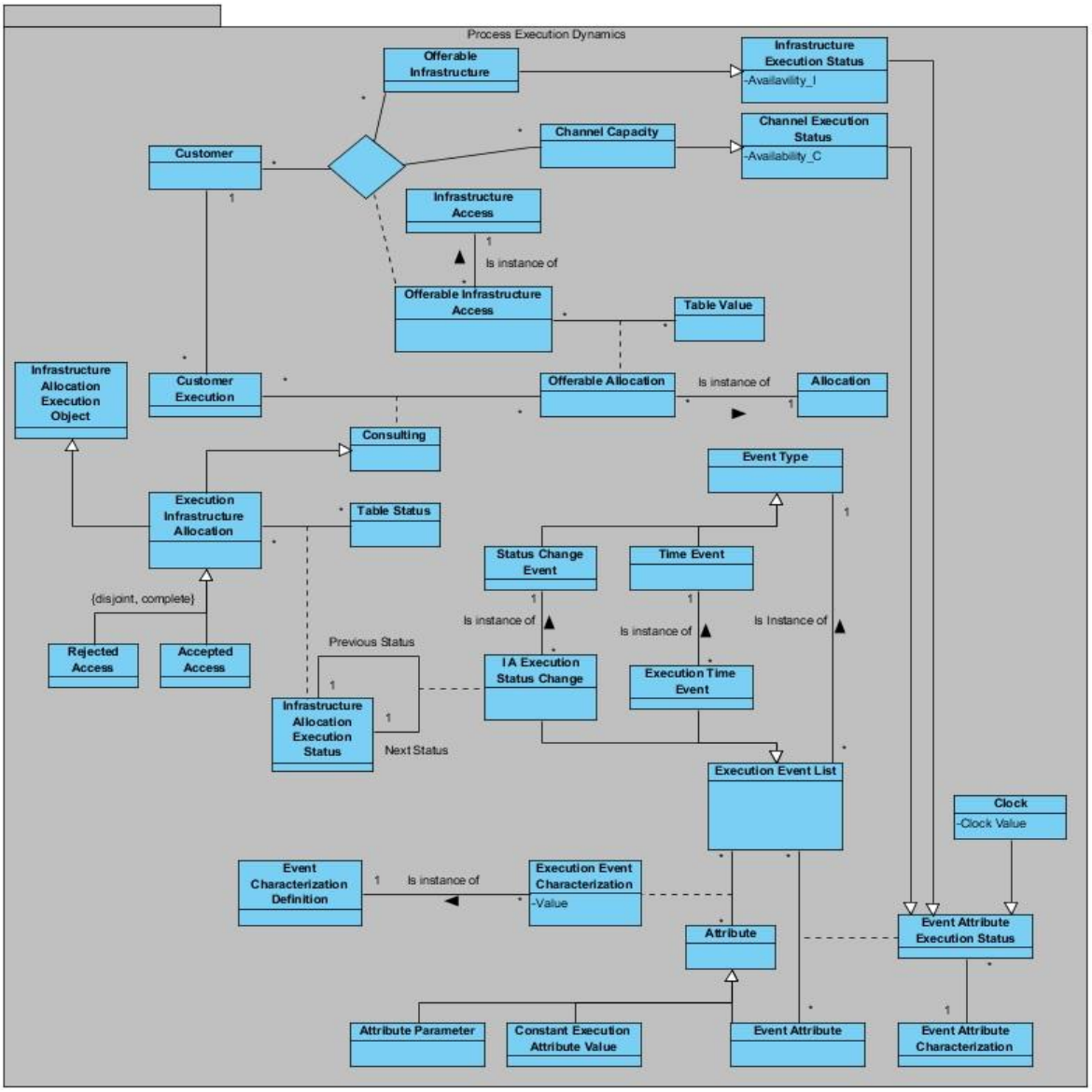


Figura 3-17. Diagrama UML de la dinámica del Proceso de Ejecución (Process Execution)

En esta lista de eventos, puede haber tanto eventos de tiempo (Execution Time Event) como eventos de cambio de estados (IA Execution Status Change), que serán instancias de Time Event y Status Change Event respectivamente, ya que en el nivel de ejecución los eventos son totalmente concretos. Es decir, los eventos suceden en un tiempo concreto de la ejecución, a un objeto concreto y en unas condiciones concretas. Además, cada evento estará caracterizado por el valor que toma en ese momento el atributo que causa dicho evento, y dicha caracterización se refleja en la clase caracterización de eventos de ejecución (Execution Event Characterization). Por ejemplo, en un determinado momento de la ejecución, una asignación formada por un cliente normal concreto que accede a reservar una determinada habitación simple, a través de una canal web concreto, y que cuesta un precio concreto, falla la conexión y pasa al estado Failed Access (Acceso Fallido) porque el atributo asociado Failed Access Probability (Probabilidad de fallo de acceso) adquiere un valor verdadero. Este evento concreto, y el valor "verdadero" que toma el atributo son registrados.

Como ya se explicó en el apartado de atributos de la instancia, cada atributo puede ser o un parámetro de atributo, o un atributo de ejecución constante, o un atributo de

evento dinámico. Respecto a los atributos de evento dinámicos, pueden cambiar de valor durante la simulación. Por ello, la ocurrencia de cualquier evento puede desencadenar el cambio de valor de un/os determinado atributo de evento. La clase Estado de ejecución de atributos de evento (Event Attribute Execution Status) contiene el valor de cada atributo de evento en cada momento de la simulación, pudiendo cambiar el valor de cada atributo con la ocurrencia de cualquier evento. Los cambios de valor de cada atributo de evento sucederán de acuerdo a la caracterización de atributos de evento, definida en la instancia del proceso, y que establece qué eventos producen el cambio y cómo varía el valor de cada atributo de evento. Por ello, una caracterización de un evento de atributo puede tener muchas ocurrencias, es decir, que el atributo correspondiente sufra el mismo cambio varias veces durante la simulación.

Destacar que tanto el reloj de la ejecución (Clock), como el estado de ejecución de los canales (Channel Execution Status), como el estado de ejecución de las infraestructuras (Infrastructure Execution Status), son subtipo del estado de ejecución de atributos de evento, como refleja la relación en el diagrama UML. El reloj de la ejecución va avanzando de acuerdo al sistema de avance del tiempo al siguiente evento, ya explicado, y toma valores por medio del atributo Value (Valor) del clock (reloj). Para cerrar el círculo, la disponibilidad tanto de canales como de infraestructuras de ejecución, irá variando en función de los eventos ocurridos, partiendo de un valor inicial definido por el usuario.

Por último, recordar que la parte inicial del diagrama de estados, y que gobierna la forma en la que los distintos clientes entran al sistema, está fijada por el modelo conceptual y reflejada en la figura 3-9, del apartado 3.4.2 correspondiente al módulo de definición de estados. Esta parte inicial del árbol de estados, por los que todo cliente que entra al sistema debe pasar, será común a todas las ejecuciones de cualquier instancia y modelo.

Con esto, se concluye la explicación de todo el modelo conceptual desarrollado durante el proyecto, y que supone la base para los dos capítulos siguientes, como son la base de datos y la aplicación. Ha sido importante explicar con detalle los modelos, ya que son los cimientos del proyecto, y por lo tanto de las partes que se explicarán a continuación y que requerirán de mucha menos explicación y detalle.

CAPÍTULO 4

MODELO DE DATOS

Tras la explicación detallada de los modelos teóricos que fundamentan el proyecto, el siguiente paso nos lleva a este capítulo, el cual está dedicado en su totalidad a la descripción del modelo de datos, es decir, a la traducción de los modelos conceptuales en la base de datos. Por ello, el objetivo de este capítulo es explicar el diseño e implementación de la base de datos, que estará coordinada con la aplicación para almacenar toda la información.

Como hemos comentado en varias ocasiones, los modelos conceptuales son la base del proyecto, y el modelo de datos está construido en línea para garantizar una coherencia y servir de demostración de la robustez y validez de los modelos conceptuales. Es decir, la base de datos y la aplicación sirven para demostrar que los modelos conceptuales son viables y consistentes, además de servir como DSS (Sistema de soporte a la toma de decisiones). Por este motivo, la estructura del capítulo seguirá el mismo orden que los modelos conceptuales, es decir, explicar las distintas partes de la base de datos empezando por el nivel jerárquico Modelo de Proceso (Process Model), e iremos dedicando un apartado a cada nivel jerárquico hasta llegar al nivel de ejecución de proceso (Process Execution). Cabe aclarar, que no empezamos por el nivel de metamodelo porque desde el punto de vista de la base de datos no se implementa en este nivel debido al grado de abstracción que conlleva. Aunque el capítulo este divido por niveles de abstracción, la base de datos final resulta de la unión de todas las partes perfectamente relacionadas. Para los diagramas se empleará la herramienta IDEF1X, pero la base de datos está implementada en Microsoft Access para la base de datos relacional.

4.1 NIVEL DE MODELO DE PROCESO

En este apartado se explica el diseño de la base de datos correspondiente al nivel jerárquico de modelo de proceso (Process Model), y representa la parte inicial de toda la base de datos.

En la figura 4-1 se muestra el diagrama IDEF1X del segmento de la base de datos correspondiente a la definición del modelo de proceso. Vemos que las únicas entidades independientes, representadas por rectángulos con bordes en forma de pico, son *Infrastructure Allocation Model* e *Infrastructure Allocation Modeling Entity*. La primera es independiente porque es la entidad donde se empieza a rellenar toda la base de datos, y no tiene ninguna clave extranjera asociada. En el caso de la segunda tabla mencionada, es donde figuran las siete entidades básicas del modelo.

En primer lugar, es importante resaltar que se ha llevado a cabo un diseño e implementación de esta capa de la base de datos integrada. Es decir, puesto que un modelo de proceso y una instancia de proceso son hermanos, aunque diferenciados en el nivel de generalidad de cada uno como ya se ha explicado anteriormente, se pueden almacenar en las mismas tablas los registros correspondientes tanto a la instancia como al modelo. Por este motivo, y acorde a la implementación de una base de datos integrada en el nivel de modelo de proceso e instancia de proceso; en todas las tablas donde se almacenan registros correspondientes a la definición del modelo de proceso, el campo ID_ProcessInstance adquirirá un valor de 0, mientras que en el caso de la instancia este campo adquirirá el valor que le corresponda. Gracias a este modelo de base de datos integrada, nos ahorramos la necesidad de duplicar todas las tablas del nivel de modelo de negocio para el nivel de instancia. Además, el modelo integrado es

posible gracias a que en el modelo conceptual el modelo de negocio y la instancia son "Hermanos".

Una vez explicado esto, cada una de las entidades básicas del modelo recibe un número identificativo, del 1 al 7 y a través del campo ID_IAM_Entity (Infrastructure Allocation Model Entity) de la tabla Infrastructure Allocation Modeling Entity. Acorde al diagrama de clase del modelo de negocio explicado en el capítulo anterior, cada modelo de asignación a infraestructuras tiene las siete entidades básicas, relación almacenada en la tabla Infrastructure Allocation Model Entity.

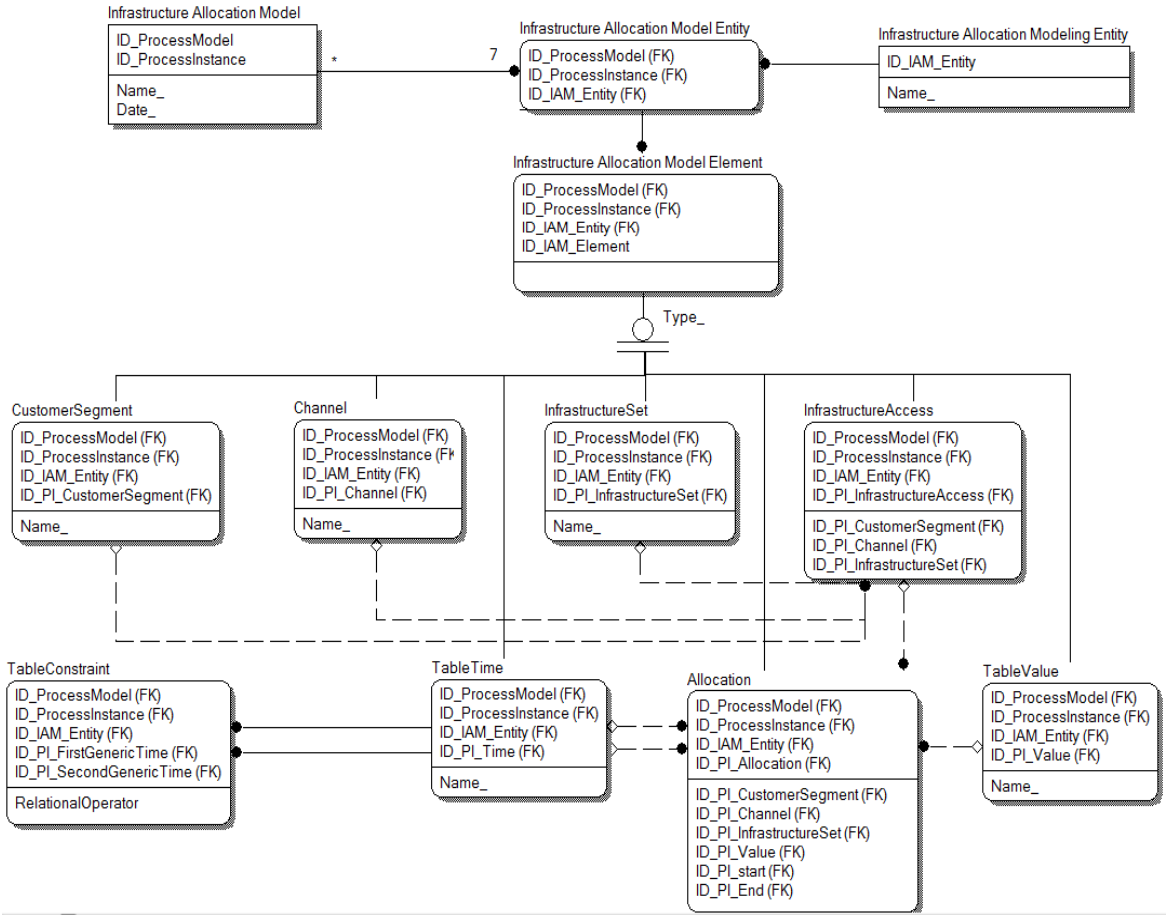


Figura 4-01. Diagrama IDEF1X del segmento de base de datos correspondiente al modelo de proceso

Para los elementos definidos pertenecientes a cada entidad, se ha decidido dedicar una tabla individual para cada una de las entidades básicas del modelo, es decir, Customer Segment, Channel, Infrastructure Set, Table Time, Table Value, Infrastructure Access, y Allocation. La tabla Infrastructure Allocation Model Element contendrá todos los registros de la definición de elementos del modelo, y el atributo Type_ describe la entidad a la que pertenece cada elemento, relacionándolo así con la tabla que corresponda. Podría pensarse que estas tablas individuales para cada entidad son innecesarias, ya que los registros ya estarán almacenados en la tabla Infrastructure Allocation Model Element, de manera que a la hora de consultar los registros se podría acudir a ésta. No obstante, la decisión de implementar todas las tablas es para facilitar la expansión de la base de datos a los siguientes niveles, y dejar abierta la posibilidad a requerimientos futuros que pudieran necesitar esta estructura.

Las siete tablas que recogen los elementos del modelo, están identificadas por la clave primaria extranjera heredada de la tabla Infrastructure Allocation Model Element, con la particularidad de que ID_IAM_Element pasa a denominarse ID_PI_CustomerSegment, ID_PI_Channel, ID_PI_InfrastructureSet, ID_PI_InfrastructureAccess, ID_PI_Value, ID_PI_Time, e ID_PI_Allocation según el caso. Añaden el campo Name_ asociado al nombre de cada elemento, salvo las entidades InfrastructureAccess y Allocation.

Respecto a la tabla InfrastructureAccess, como cada registro se corresponde con una combinación de la terna Customer Segment, Channel, e Infrastructure Set; presenta como claves no primarias las claves ajenas que identifican a cada uno de estos tres elementos. Por su parte, la tabla Allocation incorpora como claves no primarias las claves ajenas identificativas de los elementos que definen cada registro, es decir, las que hereda del elemento Infrastructure Access asociado, más el valor (ID_PI_Value) y el espacio temporal que comprende (ID_PI_Start e ID_PI_End). Las relaciones entre las tablas están perfectamente definidas de acuerdo a los modelos conceptuales, y teniendo en cuenta las exigencias del metamodelo.

La última tabla implementada es TableConstraint, y que recoge las restricciones temporales definidas para cada modelo. Con este propósito, cada registro se relaciona con dos elementos de tiempo (ID_PI_FirstGenericTime e ID_PI_SecondGenericTime) y el operador (RelationalOperator) que marca la relación, que puede ser "=" 0 "≤". Toda la clave es extranjera, y es heredada de los dos elementos de tiempo que definen el registro. En consecuencia, como cada registro está unívocamente identificado, la base de datos no permite al usuario definir una restricción entre dos elementos de tiempo que aparezcan ya relacionados en otro registro, aunque sea con distinto operador relacional.

Por último, algunas aclaraciones acerca de la implementación del sistema de tratamiento de plantillas en la base de datos. El sistema de tratamiento de plantillas recoge en este nivel modelos de proceso guardados como tal, por lo que las tablas y las características de éstas deben ser las mismas. Para poder diferenciar entre los modelos guardados de manera normal y las plantillas, nos hemos basado en la utilización de los los campos ID_ProcessModel e ID_ProcessInstance para la distinción. Hemos empleado estos campos pues tanto desde el punto de vista teórico como práctico, son campos presentes en todas las tablas, es la mejor solución. Así, el modelo guardado en la base de datos con identificativo ID_ProcessModel = -1 será la plantilla del modelo por defecto, mientras que los modelos correspondientes al resto de plantillas se guardarán con identificativos de ID_ProcessModel negativos exceptuando el -1. Respecto al campo ID_ProcessInstance, al igual que en todos los registros correspondientes a un modelo de proceso, adoptará el valor 0.

4.2 NIVEL DE INSTANCIA DE PROCESO DE NEGOCIO

En este apartado pasamos al siguiente nivel, nivel de instancia de proceso de negocio. El objetivo es presentar y explicar el diseño e implantación de la base de datos en lo correspondiente a la definición de una instancia de proceso. Los pasos a seguir para explicar todas las partes relacionadas con la instancia son, al igual que en el capítulo anterior, primero una introducción general sobre la instancia y el tratamiento de

plantillas de instancias, luego el módulo de atributos, pasando por el módulo de estados, y concluyendo con el módulo de la función objetivo.

En línea con lo explicado en el apartado anterior, ya que la instancia de proceso y el modelo del proceso son "Hermanos", los elementos definidos en el modelo de proceso son los mismos que los de la instancia, aunque se les otorgarán más características a lo largo de este nivel mediante los atributos, estados, etc. Debido a esto, y a la implementación integrada (Modulo – Instancia) de la base de datos, las tablas reflejadas en la figura 4-1 del apartado anterior serán también válidas para la instancia.

En el caso de la instancia, las tablas de la figura 4-1 se rellenaran con los mismos registros que el modelo salvo el campo ID_ProcessInstance, que en el caso del modelo valía cero y ahora tomará un valor desde uno hacia arriba según corresponda. De esta manera se aprovecha el modelo conceptual "Unificado", explicado en capítulo anterior, para implantar la base de datos integrada hasta este punto.

Respecto al sistema de tratamiento de plantillas instancia, también se han empleado los campos ID_ProcessModel e ID_ProcessInstance para diferenciar las instancias guardadas como plantillas de las instancias normales. A efectos de registros de base de datos, a lo largo de toda la instancia sólo estos dos campos serán diferentes entre una instancia guardada normalmente y una plantilla. Para las plantillas de las instancias, el campo ID_ProcessModel siempre valdrá cero, y para las instancias normales este campo valdrá el número correspondiente al identificativo del modelo de proceso asociado. En cuanto al campo ID_ProcessInstance, valdrá cero para guardar los registros pertenecientes al modelo de referencia que servirá para guardar las plantillas de instancias. Adoptará el valor -1, en todas las tablas, para guardar los registros de la plantilla de la instancia guardada por defecto, y para el resto de plantillas de instancias adoptará valores negativos distintos del -1. El campo ID_ProcessInstance valdrá números positivos por encima del cero para las instancias normales de los modelos de proceso. En la tabla 4.1 se muestra un resumen de los valores que adoptan los campos.

Valor del Campo		Concepto
ID_ProcessModel	ID_ProcessInstance	
Valores positivos desde 1 en adelante	0	Modelos de Proceso definidos
-1	0	Plantilla por defecto de Modelo de Proceso
Valores negativos excepto el -1	0	Resto de Plantillas de Modelos de Proceso
0	0	Modelo de Proceso de referencia para tratar las plantillas de las instancias
Valor correspondiente al Modelo de Proceso	Valores positivos desde 1 en adelante	Instancias de los Modelos de Proceso
0	-1	Plantilla de la instancia por defecto
0	Valores negativos excepto el -1	Resto de Plantillas guardadas de instancias

Tabla 4-1. Valores asignados a los campos ID_ProcessModel e ID_ProcessInstance en la base de datos

4.2.1 MÓDULO DE DEFINICIÓN DE ATRIBUTOS

En este apartado se explica el diseño e implementación del segmento de base de datos correspondiente al módulo de atributos de la instancia.

En la figura 4-2 se muestran las entidades (tablas) y las relaciones entre ellas que configuran la parte de la base de datos asociada al módulo de atributos de la instancia. La parte inicial del diagrama IDEF1X, las cuatro tablas superiores, representan la parte ya explicada y perteneciente a los registros de las instancias en cuanto a las entidades básicas y elementos se refiere.

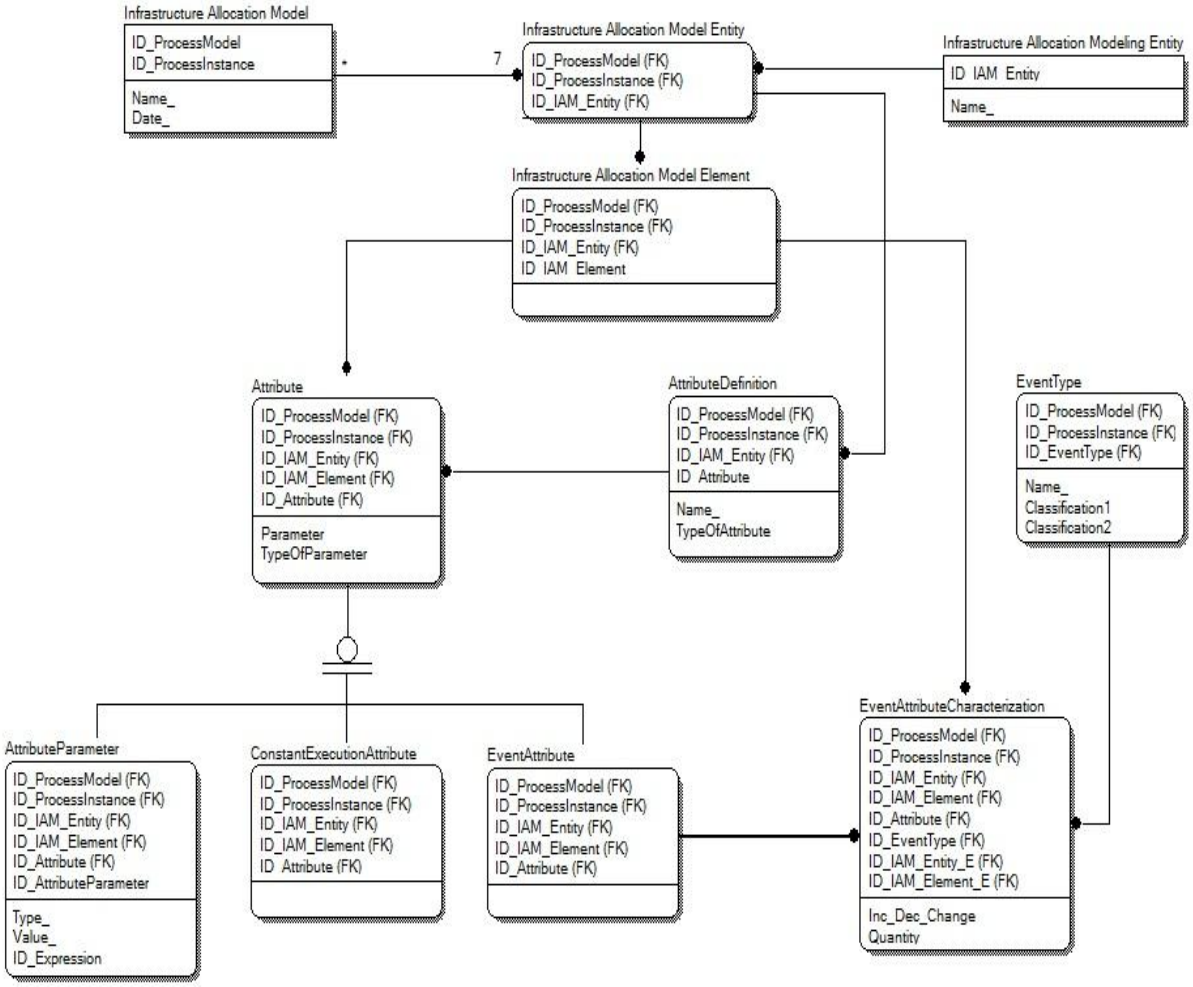


Figura 4-02. Diagrama IDEF1X del segmento de base de datos correspondiente al modelo de Atributos

Como se explicó en los modelos conceptuales, cada entidad básica puede tener muchos atributos asociados, con lo que la tabla AttributeDefinition recogerá todos los atributos definidos pertenecientes a cada entidad. Esta tabla hereda en su clave principal los tres primeros campos como claves extranjeras, e incorpora el campo ID_Attribute que identificará de forma unívoca a los atributos de una entidad concreta definidos. Además, el campo Name_ almacena el nombre y el campo TypeOfAttribute el tipo de atributo (Constant, Boolean, Distribution, Single Probability, o Piecewise Probability).

De cada atributo definido para una entidad, se asociará con cada uno de los elementos pertenecientes a dicha entidad. Es decir, se registrará un atributo por elemento que pertenezca a la entidad, ocurriendo lo mismo para las siete entidades. Esto se refleja en la base de datos en la tabla Attribute. Esta tabla tiene como clave principal todo claves extranjeras, de forma que hereda todas las claves principales de Attribute Definition, y de la tabla Infrastructure Allocation Model Element hereda la clave ID_IAM_Element que aporta los identificativos de todos los elementos de la instancia

definidos. Como claves no primarias incorpora el campo Parameter que indica si es parámetro o variable del problema, y el campo TypeOfParameter que indica si es estático o dinámico.

Acorde a los modelos conceptuales, los atributos pueden ser estáticos, en cuyo caso se registrarán en la tabla AttributeParameter, o dinámicos, entre los que se distinguen atributos dinámicos de ejecución constante, almacenados en la tabla ConstantExecutionAttribute, y atributos de evento, almacenados en la tabla EventAttribute. Con esto, cada atributo se clasifica en una de estas tres categorías y se registra en la tabla correspondiente.

Por un lado tenemos la tabla AttributeParameter, que hereda todos los campos de la clave principal de Attribute, e incorpora como clave propia en la clave primaria el campo ID_AttributeParameter. Este campo distingue entre los diferentes tipos de parámetros de atributo que se vinculan a cada atributo dependiendo del tipo de éste (Constant Value, Probability Value, Distribution Parameter 1, Logical Value, etc). En la clave no primaria figuran los campos Value_ para almacenar el valor de cada parámetro de atributo, el campo tipo texto Type_ para almacenar el tipo de parámetro de atributo, y el campo ID_Expression que se ha añadido para futuros desarrollos que puede sufrir el proyecto, y cuyo propósito se explica en la parte del capítulo de conclusiones que trata sobre futuros desarrollos. En resumen, este último campo está pensado para asociar el valor de un parámetro de atributo a una expresión matemática.

Los atributos dinámicos de ejecución constante se almacenan en la tabla ConstantExecutionAttribute, compuesta en su clave principal de los campos heredados como claves extranjeras de la tabla Attribute. Al ser atributos dinámicos no reciben valor en este nivel.

Por último tenemos los atributos de evento dinámicos, que se registran en la tabla EventAttribute, y que al igual que en el caso anterior contiene en su clave principal las claves extranjeras heredadas de la tabla Attribute. De la misma manera, estos atributos no reciben valor en este nivel, pero sí se caracteriza el cambio de valor que pueden sufrir a lo largo de las ejecuciones.

Esta caracterización de los atributos de evento dinámicos, en la que se especifica qué cambios pueden sufrir durante la ejecución, se almacena en la tabla EventAttributeCharacterization. En esta tabla se registra qué elemento del modelo desencadena un cambio en el valor de qué atributo al experimentar un determinado evento. Por ello, la tabla está relacionada con las tablas EventAttribute, de la que hereda como claves extranjeras ID_ProcessModel, ID_ProcessInstance, ID_IAM_Entity, ID_IAM_Element, e ID_Attribute; con la tabla Infrastructure Allocation Model Element, de la que hereda las claves ID_IAM_Entity_E e ID_IAM_Element_E, y finalmente la tabla EventType, de la que hereda la clave ID_EventType. Todas estas claves extranjeras conforman la clave primaria de EventAttributeCharacterization; y en la clave no principal incluye el campo Inc_Dec_Change, de tipo texto y que registra el tipo de cambio de valor que se produciría, y el campo Quantity, que especifica la cantidad del cambio si corresponde o un valor nulo en caso contrario.

4.2.2 MÓDULO DE DEFINICIÓN DE ESTADOS

Pasamos ahora al módulo de estados de la instancia de proceso. Es el segundo módulo en la definición de la instancia, y explicaremos la el segmento de la base de datos orientada a este módulo.

El diseño de la parte de la base de datos correspondiente a este módulo, al igual que toda la base de datos, se ha llevado a cabo de acuerdo a los modelos conceptuales. En concreto, en este caso de acuerdo al diagrama UML de la figura 3-8. El diagrama IDEF1X de la figura 4-3 muestra la traducción de dichos modelos conceptuales, pertenecientes al módulo de estados, en la base de datos.

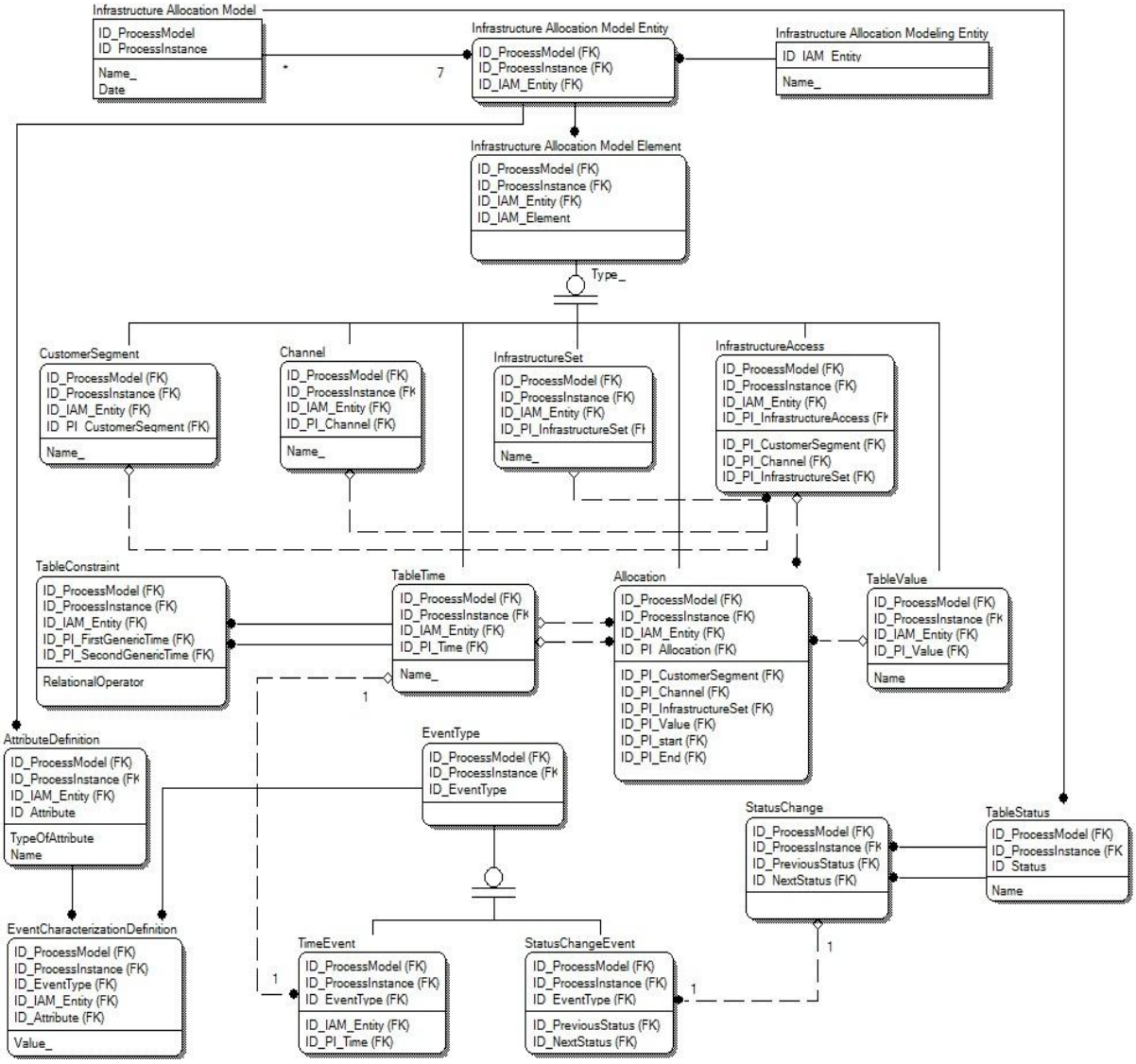


Figura 4-03. Diagrama IDEF1X del segmento de base de datos correspondiente al modelo de Estados

Se ha considerado que cada una de las entidades que figuran en el diagrama UML nombrado anteriormente, requieren una tabla propia en la base de datos. Por lo tanto, en este módulo disponemos de las tablas TableStatus, StatusChange, StatusChangeEvent, TimeEvent, EventType, y EventcharacterizationDefinition.

La primera tabla que se rellena al comenzar a definir la dinámica de una instancia de proceso de negocio es TableStatus. En cada instancia se definen una serie de estados, por lo que pueden pasar los clientes durante la ejecución del proceso. Por esto, la clave principal de la tabla TableStatus está compuesta por los campos ID_ProcessModel e ID_ProcessInstance heredados de Infrastructure Allocation Model, y del campo propio ID_Status, que identifica unívocamente a cada estado definido. En la clave no primaria se almacena el nombre de cada estado mediante el campo Name_. Al igual que en casos anteriores, el guión bajo que sigue a "Name" es debido a que en el lenguaje SQL esta palabra es clave. Ocurre lo mismo con los campos denominados Type_.

Para pasar de un estado a otro el usuario debe definir los posibles cambios de estado que pueden ocurrir. Esto se almacena en la tabla StatusChange, definiendo el estado previo y el siguiente estado que definen el cambio de estado. Cada registro de esta tabla estará relacionado con dos registros de la tabla TableStatus, y por ello la relación es la que figura en el diagrama de la figura 4-3. La clave principal es totalmente extranjera y heredada de la tabla tableStatus, donde el campo ID_Status se transforma en ID_PreviousStatus para el estado previo, e ID_NextStatus para el siguiente estado.

Ya que los eventos son o bien cambios de estado o bien eventos de tiempo, de la tabla StatusChange nace la tabla StatusChangeEvent con una relación uno a uno. Esta relación es porque cada cambio de estado es un evento, y por lo tanto cada registro de la tabla StatusChange pasa automáticamente a la tabla StatusChangeEvent para asociarse con los identificativos de eventos. Por ser una relación uno a uno, la tabla StatusChangeEvent hereda todas las claves que forma la clave principal de StatusChange, aunque ID_PreviousStatus e ID_NextStatus pasan a formar parte de la clave no principal de StatusChangeEvent. Ademas, incorpora en la clave principal el campo identificativo ID_EventType, pasa asociarse con un evento.

Por otro lado tenemos los eventos de tiempo, teniendo en cuenta que cada evento de tiempo está marcado por los elementos de tiempo definidos en el modelo. Por este motivo, la tabla TimeEvent, al igual que en el caso anterior, nace de una relación uno a uno con la tabla TableTime, que almacena los elementos de tiempo del modelo. De la misma manera que en el caso anterior, TimeEvent hereda las claves de TableTime, siendo ID_IAM_Entity e ID_PI_Time parte de la clave no primaria de TimeEvent.

Con esto, y como hemos comentado, los eventos de la instancia están formados por todos los cambios de estado definidos, y por los eventos de tiempo que marcan los elementos de tiempo. Así, la tabla EventType recoge todos los identificativos de evento que existen en la dinámica de la instancia a través del campo ID_EventType que define unívocamente a cada evento. Esta tabla se asocia con las tablas TimeEvent y statusChangeEvent por medio de una relación de clasificación completa.

La última tabla es EventCharacterizationDefinition, y es donde se almacena la caracterización de cada evento de la instancia. Cada evento esta caracterizado por uno o dos atributos y por el valor que éstos deben adoptar para la ocurrencia del evento. En consecuencia, la tabla EventCharacterizationDefinition hereda en su clave principal por un lado los campos ID_ProcessModel, ID_ProcessInstance, e ID_EventType de la tabla EventType, y por otro lado los campos ID_IAM_Entity e ID_Attribute de la tabla AttributeDefinition. Estas cinco claves extranjeras componen la clave primaria, y el valor que deben adoptar los atributos se almacena en el campo propio Value_ y que forma parte de la clave no principal.

4.2.3 MÓDULO DE DEFINICIÓN DE LA FUNCIÓN OBJETIVO

El modelo de datos del módulo de la definición de la función objetivo de la instancia es el tema de este apartado. Es la última parte de la base de datos con respecto al nivel de instancia de proceso. Para explicar dicho modelo de datos nos basamos en el diagrama IDEF1X elaborado con ERWin que aparece en la figura 4-4, y que representa el segmento de la base de datos correspondiente al módulo de función objetivo.

Puesto que a la hora de diseñar e implementar la base de datos nos basamos siempre en los modelos conceptuales, en este caso nos basamos en el diagrama UML mostrado en la figura 3-11del capítulo anterior. Se ha decidido que cada una de las entidades presentes en dicho diagrama UML requieren de una tabla independiente en la base de datos. Por ello, en este punto incluimos las tablas ObjectiveFunction, Lgic Expression, Optimization Expression, Mathematical Expression, Term, Compound Term, Basic Term, Operation, Multiple Operation, y Basic Component.

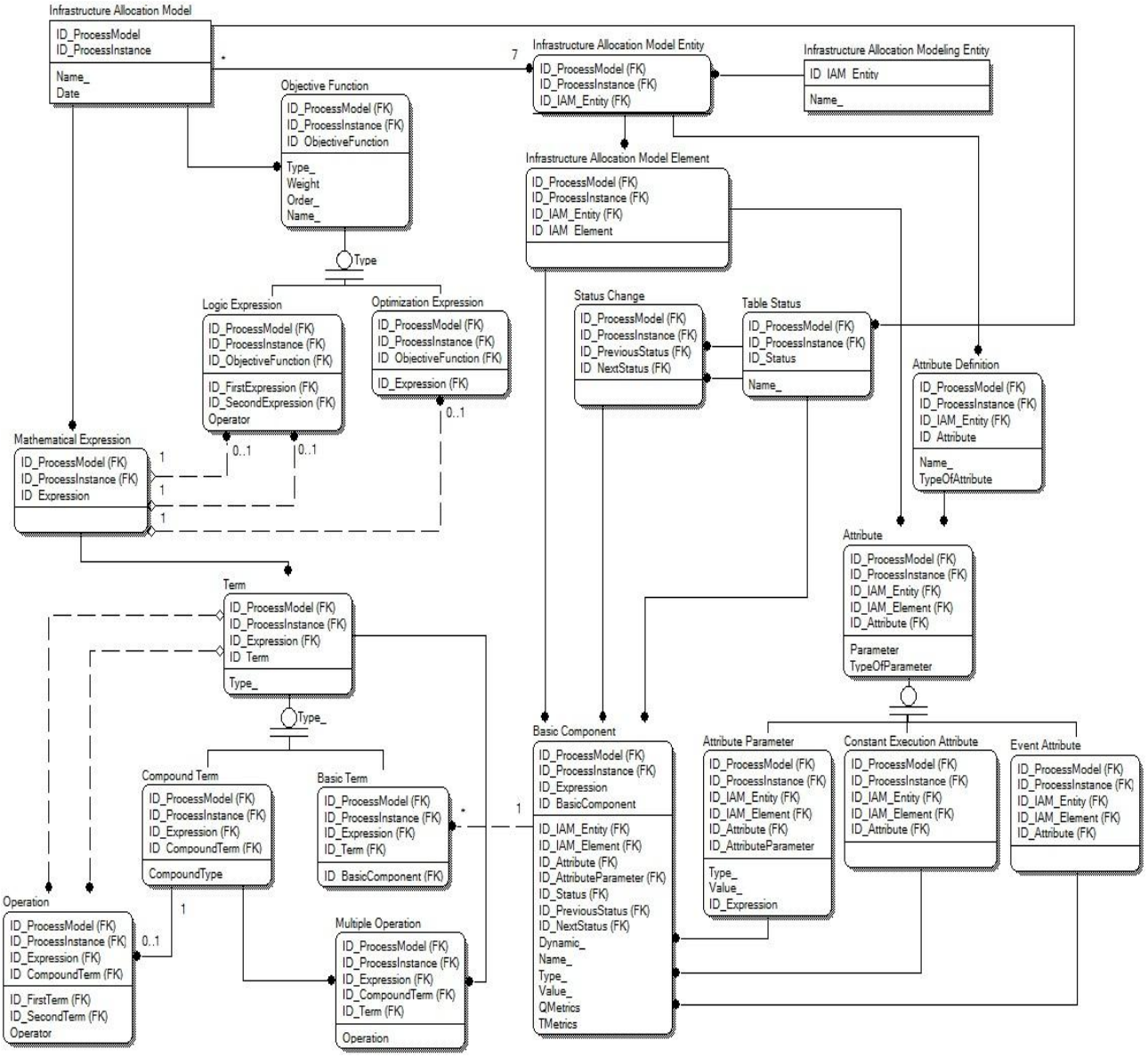


Figura 4-04. Diagrama IDEF1X del segmento de base de datos correspondiente al modelo de Función Objetivo

Este segmento de la base de datos comienza teniendo en cuenta la consideración de que cada instancia del modelo de asignación de infraestructuras puede tener muchas funciones objetivo. Además, puesto que las funciones objetivo están compuestas por expresiones matemáticas (Mathematical Expression), cada instancia del proceso puede tener muchas expresiones. Por lo tanto, las dos primeras tablas que se rellenan en este segmento de la base de datos son ObjectiveFunction y Mathematical Expression.

La tabla ObjectiveFunction tiene en su clave principal las dos claves heredadas de Infrastructure Allocation Model, ID_ProcessModel e ID_ProcessInstance, y la clave propia ID_ObjectiveFunction que identifica unívocamente a cada una de las funciones definidas. No menos importante es la clave no primaria, que está formada por los campos propios Type_, que indica el tipo de función objetivo (Maximización, minimización, o lógica), Weight, que determina el peso que tiene cada función, Order_, para especificar qué orden de prioridad tiene cada función respecto al resto, y Name_, que recoge el nombre asignado a cada una. Como ocurre con la asignación de nombres a los campos en toda la base de datos, los campos cuyo nombre termina con un guión bajo es debido a evitar conflictos con las palabras clave del lenguaje SQL de la base de datos.

Acorde a los modelos conceptuales, cada función objetivo se puede clasificar en funciones de optimización, ya sea de maximización o de minimización, y funciones lógicas, en las que se dos expresiones matemáticas se relacionan mediante un operador para expresar una restricción o condición. Por lo tanto, para registrar esta clasificación a través del atributo Type_, contamos con las tablas Logic Expresion (Expresión lógica) y Optimization Expression (Expresión de optimización).

La tabla Optimization Expression tiene toda la clave primaria extranjera, pues los tres campos ID_ProcessModel, ID_ProcessInstance, e ID_ObjectiveFunction son heredados de la tabla Objective Function. Además, para asociarse al identificativo de la expresión matemática que define cada función de optimización, se presenta el campo de la clave no primaria ID_Expression, que es heredado de la tabla Mathematical Expression.

La tabla Logic Expression, al igual que en caso anterior tiene toda la clave primaria compuesta de los tres campos heredados de la tabla Objective Funcction. Pero en este caso la clave no primaria es diferente, esto es debido a que cada función lógica está compuesta de dos expresiones matemáticas relacionadas por un operador. Con esto, la clave extranjera ID_FirstExpresion recoge el identificativo de la primera expresión de la función, e ID_SecondExpression recoge la segunda expresión, siendo el campo Operator propio y que especifica el operador relacional empleado ($<, \leq, =, \geq, >$).

Respecto a la tabla Mathematical Expression, como hemos comentado anteriormente es junto con Objective Function la primera en rellenarse. Incorpora en su clave principal los campos ID_ProcessModel e ID_ProcessInstance heredados de Infrastructure Allocation Model, y la clave propia ID_Expression que identifica unívocamente cada expresión matemática. Una expresión matemática puede ser una función de optimización, y relacionarse con la tabla Optimization Expression, y ser una de las dos expresiones de una función lógica, y relacionarse con la tabla Logic Expression. Cada registro de la tabla Logic Expression se relaciona con dos registros de la tabla Mathematical Expression.

Avanzando en el diagrama, cada expresión matemática está compuesta por muchos términos, recogidos en la tabla Term. Tabla cuya clave primaria está formada por tres campos heredados de Mathematical Expression, que son ID_ProcessModel, ID_ProcessInstance, e ID_Expression, y una clave propia ID_Term que identifica a cada término definido. Gracias al campo Type_ de la clave no primaria, cada término se clasifica en básico (Basic Term) o compuesto (Compound Term).

La tabla Compound Term recoge los términos que son compuestos, heredando toda la clave primaria de la tabla Term, e incluyendo en la clave no primaria el campo CompoundType que diferencia entre términos compuestos simples y múltiples. Los términos compuestos simples, son dos términos relacionados por un operador, característica que se almacena en la tabla Operation. Los términos compuestos múltiples son sumatorios o productorios de varios términos, y se recogen en la tabla Multiple Operation.

La tabla Operation presenta como clave principal los cuatro campos heredados de la tabla Compound Term. La clave no primaria expresa los dos términos en los que se descompone el término compuesto, correspondientes a los campos ID_FirstTerm (Primer término) e ID_SecondTerm (Segundo término) heredados de la tabla Term, y el operador que los relaciona, especificado en el campo propio Operator. Por lo tanto, cada registro de esta tabla estará relacionado con dos registros de la tabla Term.

En la tabla Multiple Operation se recogen todos los términos que forman parte de un sumatorio o productorio, es decir, de un término compuesto. La clave principal es totalmente extranjera y compuesta por los campos ID_ProcessModel, ID_ProcessInstance, ID_Expression, e ID_CompoundTerm, heredados de la tabla Compound Term y que identifican el término compuesto en definición, y el campo ID_Term, heredado de la tabla Term y que sirve para asociar el término compuesto con todos los términos en los que se descompone. El campo propio Operation de la clave no primaria es para expresar si se trata de un sumatorio o un productorio (0 para sumatorio y 1 para productorio).

La tabla Basic Term por su parte, sirve para almacenar los términos que son básicos. En línea con los modelos conceptuales, cada término básico está relacionado con un componente básico (Basic Component), y varios términos básicos pueden hacer referencia al mismo componente básico, de aqui la relación con la tabla Basic Component. Por esto, la clave principal de Basic Term es totalmente heredada de Term, y la clave no principal añade el campo ID_BasicComponent heredado de la tabla Basic Component y que sirve de enlace con el componente básico asociado.

Por último, tenemos la tabla Basic Component. Para su diseño e implementación nos hemos basado en el diagrama UML de la figura 3-12, explicada en el capítulo anterior. En esta figura se presenta la clasificación detallada de un componente básico, y para la base de datos se ha considerado tres grandes grupos: Componente de cambio de estado, componente de estado, y componente de atributo. Por esto, la tabla se relaciona con StatusChange en caso de ser un componente de cambio de estado, TableStatus en caso de ser un componente de estado, o las tres tablas correspondientes a los tipos de atributo en caso de ser un componente de atributo, AttributeParameter, Const Execution Attribute, y Event Attribute. Además, se relaciona con la tabla Infrastructure Allocation Model element para, en caso de ser un componente de estado o de cambio de estado, especificar el elemento del modelo asociado a dicho estado o cambio de estado.

Se podría haber optado por implementar una tabla individual para cada tipo de componente básico, pero para facilitar y unificar el registro de información, se ha optado por implementar sólo la tabla Basic Component, por lo que cada registro estará asociado sólo a uno de los tipos ya explicados. La clave principal incluye tres campos que identifican la expresión a la que va asociado el componente, ID_ProcessModel, ID_ProcessInstance, e ID_Expression, y el campo propio ID_BasicComponent, que identifica unívocamente a cada componente básico y que se relaciona con la tabla Basic Term. La clave no primaria, dependiendo de qué tipo de componente sea, hereda las claves ID_IAMEntity e ID_IAM_Element de Infrastructure Allocation Model Element, ID_Attribute de una de las tres tablas de atributos dependiendo del tipo, ID_AttributeParameter de Attribute Parameter, ID_Status de Table Status, y ID_PreviousStatus e ID_NextStatus de la tabla StatusChange. El campo Dynamic_ indica si el componente es dinámico o bien un número, y si es dinámico su nombre se almacena en Name_. Type_ almacena el tipo de componente en base a los tres tipos mencionados. Value_ almacena el valor en caso de ser un número. QMetrics y TMetrics almacenan la medida utilizada en caso de ser un componente de atributo.

4.3 NIVEL DE CONJUNTO DE EJECUCIONES DEL PROCESO

Una vez explicada la parte de la base de datos referente a la definición del modelo de proceso e instancia del proceso de negocio, avanzamos al cuarto nivel jerárquico, o nivel de conjunto de ejecuciones (Execution Set). El objeto de este apartado es explicar el diseño e implementación del segmento de la base de datos correspondiente a este nivel, basándonos en el diagrama IDEF1X de la figura 4-5 que representa la parte de la base de datos de este nivel.

Para la traducción de los modelos conceptuales a la base de datos nos hemos guiado por el diagrama UML de la figura 3-13, correspondiente al nivel de Execution Set. Recordar que este nivel sirve para asignar valor a las variables declaradas del problema, así como a los atributos dinámicos. Los atributos dinámicos de ejecución constante reciben un valor, que permanecerá durante las diferentes ejecuciones, y los atributos de evento dinámico reciben un valor inicial, que podrá ir cambiando a lo largo de cada ejecución. En cuanto a la decisión de qué tablas deben ser implementadas en la base de datos de acuerdo a los modelos conceptuales, se han incluido las tablas IA Process Execution Set, ExecutionSetVariable, Event Attribute Initial value, y Constant Execution Attribute Parameter.

La primera consideración importante es que cada instancia de proceso puede tener muchos conjuntos de ejecuciones. Por esto, la primera tabla que toma valor es IA Process Execution Set (Conjunto de ejecuciones del proceso de asignación de infraestructuras). La clave principal está formada por los campos ID_ProcessModel e ID_Processinstance heredados de Infrastructure Allocation Model, y la clave propia ID_ProcessExecutionSet que identifica a cada conjunto de ejecuciones. En la clave no principal se almacena el nombre asignado a cada conjunto de ejecuciones mediante el campo propio Name_, y el número de simulaciones/ejecuciones que el usuario desea definir, almacenado en el campo propio NºReproductions.

Los valores asignados a las variables del problema se almacenan en la tabla ExecutionSetVariable. Cada conjunto de ejecuciones puede disponer de muchas

variables, y se caracteriza por el valor que éstas adoptan. La clave principal de esta tabla es en su totalidad extranjera, y compuesta por los tres campos ID_ProcessModel, ID_ProcessInstance, e ID_ProcessexecutionSet heredados de la tabla IA Process xecution Set, y los campos ID_IAM_Entity, ID_IAM_Element, e ID_Attribute provenientes de la tabla Attribute, que es en la que se almacena las variables, que define el usuarios, a través del campo Parameter que adopta el valor NO (Variable). El campo propio Value_ de la clave no principal almacena el valor que se le asigna a cada variable, teniendo en cuenta que sólo pueden ser variables del problema atributos de tipo Constant (Constante) o Boolean (Booleano).

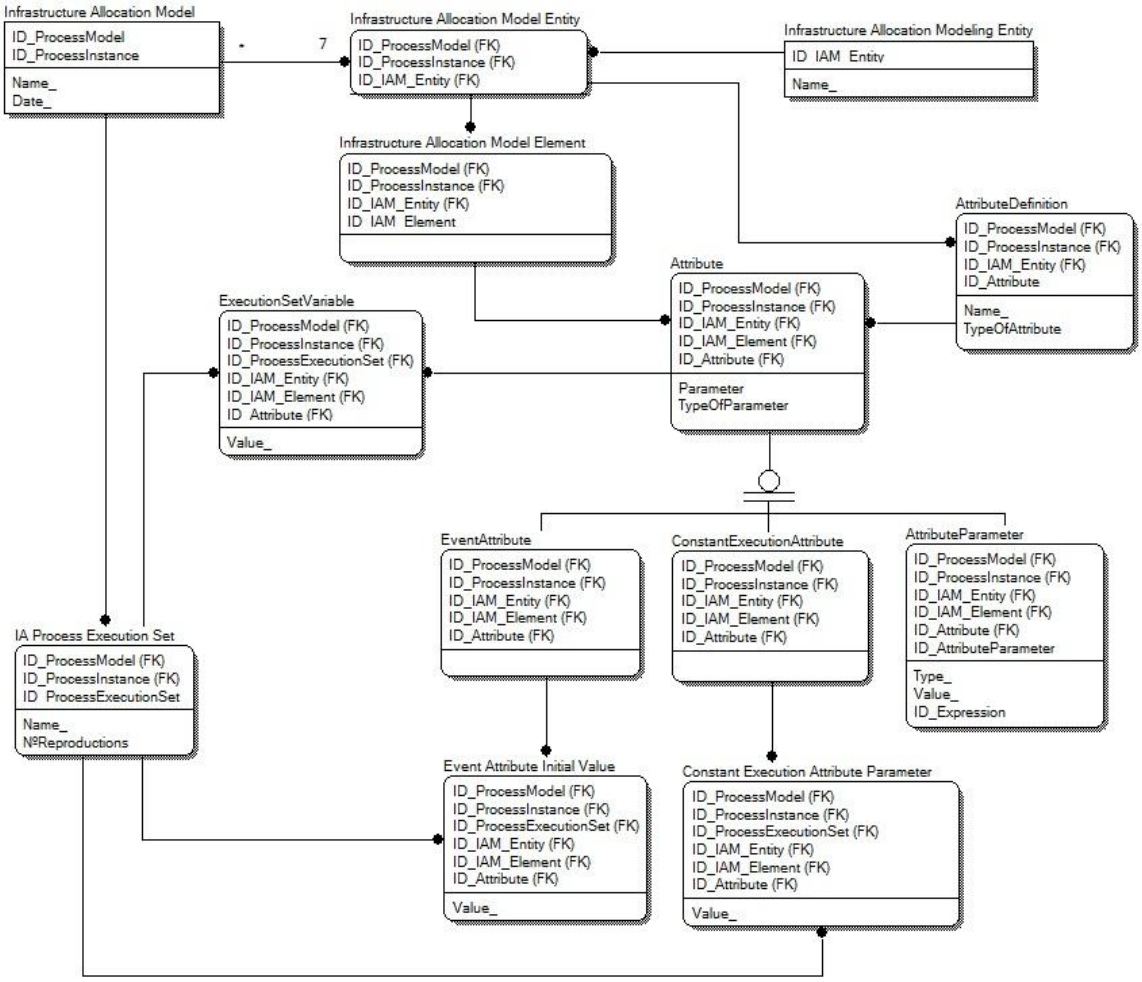


Figura 4-05. Diagrama IDEF1X del segmento de base de datos correspondiente al nivel de Execution set

En cuanto a la tabla Event Attribute Initial Value, almacena los valores iniciales de los atributos de evento dinámico. De nuevo, todos los campos de la clave principal son extranjeros, siendo ID_PorcessModel, ID_ProcessInstance, e ID_ProcessExecutionSet heredados de la tabla IA Process Execution Set, pues cada conjunto de ejecuciones puede disponer de muchos atributos de evento, y los campos ID_IAM_Entity, ID_IAM_Element, e ID_Attribute heredados de la tabla EventAttribute, que contiene todos los atributos definidos como de evento dinámico. El campo propio Value_ de la clave no primaria almacena el valor inicial de cada atributo. Recordar que como atributos de evento dinámico sólo pueden ser declarados los atributos de tipo constante (Constant), booleano (Boolean), de probabilidad única (Single Probability), o de distribución (Distribution).

La última tabla es Constant Execution Attribute Parameter, y que almacena el valor de los atributos dinámicos de ejecución constante. Al igual que en el caso anterior, hereda toda la clave principal de la tabla IA Process Execution Set, y en este caso de Constant Execution Attribute, la cual almacena todos los atributos dinámicos de ejecución constante definidos. De la misma manera que en la tabla anterior, el campo propio Vale_ de la clave no principal almacena el valor del atributo.

4.4 EJECUCIÓN DEL PROCESO DE NEGOCIO

Llegamos en este apartado al último escalón en la jerarquía de niveles, es decir, al nivel de ejecución en el que tienen lugar las simulaciones de los procesos de asignación de infraestructuras definidos en los niveles anteriores. El objetivo de este apartado es la explicación del diseño e implementación de la parte de la base de datos correspondiente al nivel de ejecución, y que supone la parte que cierra la base de datos. Para este fin, la base es el diagrama IDEF1X de la figura 4-6, y que muestra tanto las tablas y sus campos, como las relaciones entre ellas del segmento correspondiente al nivel de ejecución.

Para el ejercicio de diseño e implementación, nos hemos basado en los modelos conceptuales de este nivel, representados mediante los diagramas UML de las figuras 3-16 (Modelo estático) y 3-17 (Modelo de la dinámica de la ejecución). En primer lugar se presentan las tablas correspondientes al modelo estático de la figura 3-16, y se ha decidido implementar una tabla para cada una de las entidades presentes en este modelo UML. Las nuevas tablas son IA Process Execution, IA Execution Object, Allocation Execution, Infrastructure Set Execution, Channel Execution, Time Execution, Value Execution, Infrastructure Access Execution, y Customer Execution.

Cada conjunto de ejecuciones puede tener muchas ejecuciones, tantas como haya definido el usuario, por lo que la tabla IA Process Execution identifica cada ejecución. Su clave principal está compuesta por los campos ID_ProcessModel, ID_ProcessInstance, e ID_ProcessExecutionSet heredados de IA Process Execution Set, y la clave propia ID_ProcessExecution.

Cada ejecución tendrá muchos objetos de ejecución, que serán los elementos del modelo definidos pero que participan en la simulación en un momento concreto. En este nivel pasan de ser elementos a objetos totalmente concretos con propiedades totalmente concretas. Estos objetos se almacenan en la tabla IA Execution Object, y serán los clientes, canales, infraestructuras, etc que forman parte de la simulación. Por ello, esta tabla tiene casi toda la clave principal extranjera más el campo propio ID_Access, que identifica unívocamente a cada objeto en función de a qué acceso de cliente esté asociado. Esta tabla por lo tanto hereda los cuatro campos de la tabla IA Process Execution, y los campos ID_IAM_Entity e ID_IAM_Element de la tabla IA Model Element, destacando que este último campo se transforma en ID_IAE_Object.

La tabla IA Execution Object contiene los registros de los objetos de las siete entidades básicas del modelo. Podría emplearse sólo la tabla IA Execution Object para almacenar y gestionar los registros correspondientes a los objetos de ejecución; no obstante, se ha decidido implementar también una tabla individual para almacenar los objetos pertenecientes a cada una de las siete entidades básicas del modelo para una

mejor y más claro almacenaje, además de para no limitar futuras posibles modificaciones o desarrollos. Es decir, las tablas Allocation Execution, Infrastructure Set Execution, Channel Execution, Time Execution, Value Execution, Infrastructure Access Execution, y Customer Execution. Cada una de estas tablas pertenece a una entidad básica diferente, motivo por el que la tabla IA Execution Object se subdivide en estas siete tablas en forma de clasificación completa. Cada tabla presenta la clave principal extranjera, heredada en su totalidad de la tabla IA Execution object, y donde el campo ID_IAE_Object se transforma según corresponda en ID_AllocationExecution, ID_ChannelExecution, etc.

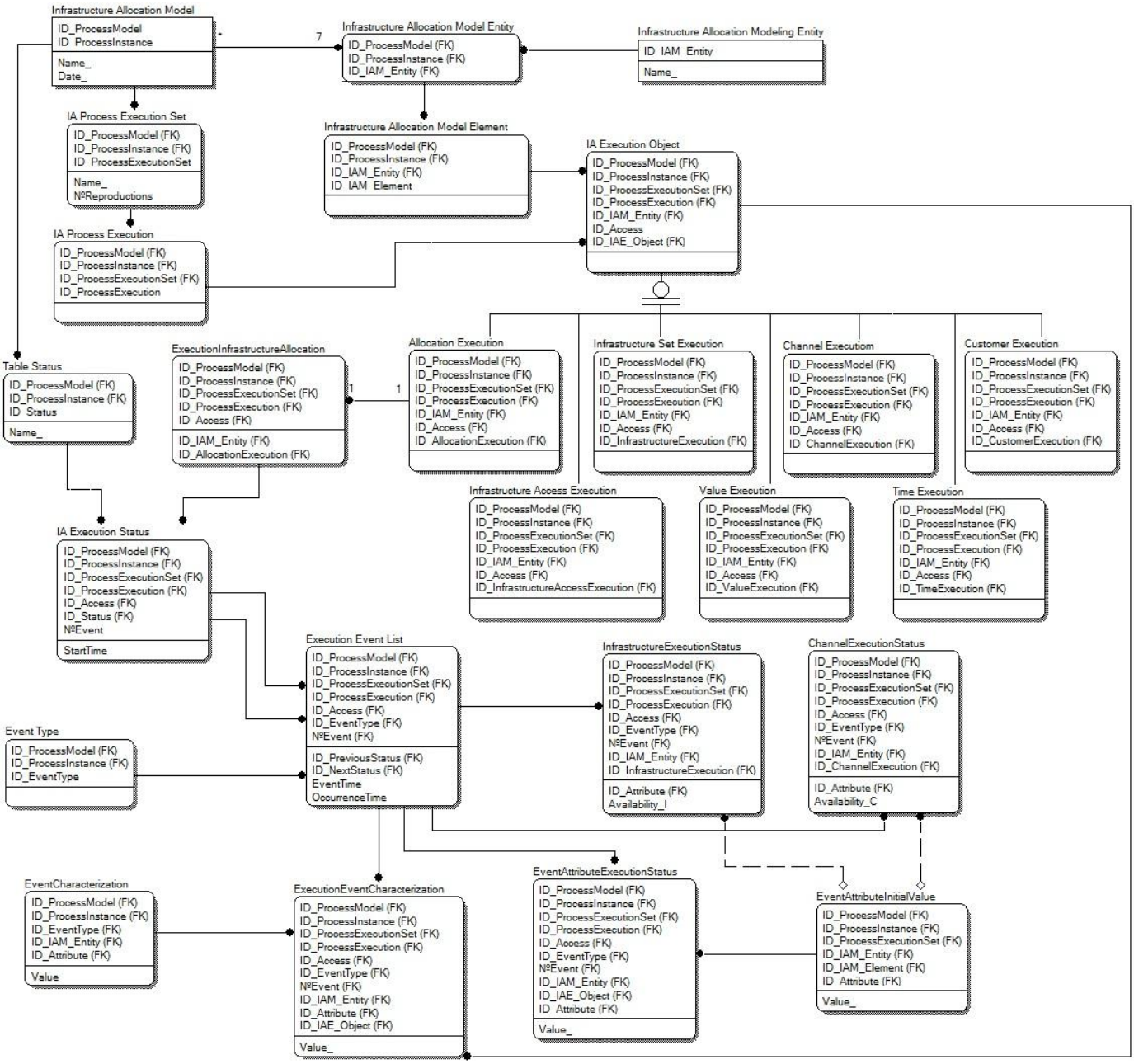


Figura 4-06. Diagrama IDEF1X del segmento de base de datos correspondiente al nivel de Ejecución

El resto de tablas de este nivel corresponden al modelo conceptual UML de la dinámica de la ejecución de la figura 3-17. Comenzamos por la tabla ExecutionInfrastructureAlloctaion, que presenta una relación uno a uno con Allocation

Execution, y es debido a que son los objetos asignación (Allocation) los que pasarán por los diferentes estados en la simulación. Esta tabla hereda todas las claves de Allocation Execution, aunque ID_IAM_Entity e ID_AllocationExecution no forman parte de la clave principal. El motivo de esta tabla es para evitar arrastrar estos dos último campos por el resto de tablas que explicaremos a continuación.

La siguiente tabla es IA Execution Status, que almacena la información relativa al paso de los diferentes objetos asignación por los diferentes estados. La clave principal contiene los cinco primeros campos heredados de la tabla ExecutionInfrastructureAllocation (ID_ProcessModel, ID_ProcessInstance, ID_ProcessExecutionSet, ID_ProcessExecution, e ID_Access), y el campo ID_Status heredado de la tabla TableStatus, que contiene todos los estados por lo que puede pasar el objeto. Además, incluye el campo propio NºEvent, que identifica la secuencia de estados que experimenta cada objeto asignación, necesario en caso de que un mismo objeto pase más de una vez por el mismo estado. La clave no primaria está formada por el campo propio StartTime, que almacena el tiempo en el que el objeto pasa a cada estado.

La siguiente tabla es fundamental en el funcionamiento de la simulación, se trata de la tabla Execution Event List (Lista de eventos de ejecución). En esta tabla se almacenan todos los eventos que tienen lugar en la simulación, y gobierna la secuencia de ejecución y los sucesos. Puesto que los eventos pueden ser de tiempo o de cambio de estado, esta tabla se relaciona con IA Execution Status y EventType. De IA Execution status hereda en la clave principal los campos ID_Processmodel, ID_ProcessInstance, ID_ProcessExecutionSet, ID_ProcessExecution, ID_Access, y NºEvent, y en la clave no principal ID_PreviousStatus e ID_NextStatus. Estos dos últimos campos marcan el cambio de estado que se produce, y cada registro de la tabla Execution Event List se relaciona con dos de la tabla IA Execution Status si el evento siempre que el evento sea de cambio de estado. Además, cada cambio de estado tiene un identificativo de evento único, marcado por el campo ID_EventType de la tabla Event Type. Por ello, el campo ID_Eventtype de la clave principal de la tabla Execution Event List es heredado de la tabla Event Type. Este campo también identifica los eventos de tiempo que tienen lugar. Finalmente, se incorporan dos campos propios en la clave no primaria, que son EventTime, que registra el tiempo en el que el evento está programado que suceda, y el campo OccurrenceTime, que almacena el tiempo en el que sucede la necesidad del evento, es decir, el momento en el que el evento se programa de cara al futuro. EventTime será siempre mayor o igual a OccurrenceTime.

La tabla ExecutionEventCharacterization almacena los valores que toman los atributos en la ocurrencia de cada evento, es decir, las condiciones en las que cada evento tiene lugar. En cuanto a la clave principal, los siete primeros campos son heredados de la tabla Execution Event List (Desde ID_ProcessModel hasta NºEvent), de la tabla EventCharacterization hereda los campos ID_IAM_Entity e ID_Attribute, y el campo ID_IAE_Object lo hereda de la tabla IA Execution Object. Con estos campos se define unívocamente el evento que tiene lugar, el objeto que lo experimenta y el atributos o atributos que lo desencadenan. El valor que adquieren dichos atributos para la ocurrencia del evento se almacenan en el campo Value_, fuera de la clave principal.

La tabla EventAttributeExecutionStatus es la encargada de almacenar en todo momento el valor que adquieren los atributos de evento dinámico, puesto que pueden cambiar a lo largo de la ejecución. Estos atributos comienzan con un valor inicial

definido por el usuario, y algunos eventos pueden dar lugar a cambios en los valores. La clave principal contiene todos los campos extranjeros, siendo los siete primeros heredados de la tabla Execution Event List (Desde ID_ProcessModel hasta NºEvent), y los campos ID_IAM_Entity, ID_IAE_Object, e ID_Attribute heredados de la tabla EventAttributeInitialValue. El campo propio de la clave no principal Value_ almacena el valor que adopta cada atributo en cada momento.

Las dos últimas tablas, InfrastructureExecutionStatus y ChannelExecutionStatus, representan casos particulares de atributos de evento (Subtipos), y aunque funcionan de la misma manera que la tabla anterior son tratadas de manera independiente. Podrían formar parte de la tabla anterior, pero por estar relacionados con los atributos fijados por el modelo Availability_I (Disponibilidad de infrasestructuras) y Availability_C (Disponibilidad de canales) respectivamente, se ha decido tratar las tablas por separado. En ambas tablas, de manera semejante al caso anterior, la clave principal está compuesta por los siete primeros campos heredados de la tabla Execution Event List (Desde ID_ProcessModel hasta NºEvent), y los campos ID_IAM_Entity e ID_InfrastructureExecution o ID_ChannelExecution, según corresponda a cada tabla, heredados de la tabla EventAttributeInitialValue. En este caso, el campo ID_Attribute heredado de la tabla EventAttributeInitialValue no pertenece a la clave principal, pues al ser una tabla relacionada con un único atributo (Availability_I o Availability_C), el campo se repetiría para todos los registros. El otro campo de la clave no principal es Availability_I o Availability_C, según corresponda, y almacena el valor del atributo en todo momento de la ejecución. Este campo muestra el número de infraestructuras disponibles y de canales disponibles de cada tipo y en cada momento.

CAPÍTULO 5

APLICACIÓN

Ya se han explicado dos de los tres bloques fundamentales que construyen el proyecto. El primero es los modelos conceptuales explicados en el capítulo 3, luego el modelo de datos explicado en el capítulo 4, y finalmente en este capítulo se va a explicar detalladamente el diseño e implementación de la aplicación. La aplicación se llama ***RM Modeler 2.0***, y está compuesta por diferentes formularios mediante los que el usuario va definiendo las propiedades y características de cada proceso de negocio. Por lo tanto, en este capítulo se describirá cada uno de los formularios, así como las razones que nos han llevado al diseño actual de la herramienta y las posibilidades que ésta presenta y que pueden ser explotadas por el usuario. Debido a que la aplicación consta de muchas ventanas de diálogo que aparecen al seleccionar diferentes opciones, no todas podrán ser explicadas, pero sí su funcionalidad. Como ya se expuso en el capítulo introductorio, el objetivo de la aplicación es ser una herramienta flexible, rápida e intuitiva a lo largo del proceso de definición, y coherente con la base de datos y modelos conceptuales.

La aplicación está coordinada con la base de datos explicada en el capítulo anterior, y el orden de los formularios es secuencial, es decir, se van mostrando al usuario de forma secuencial desde los formularios destinados a la definición del modelo de proceso, y profundizando en los siguientes niveles jerárquicos y su definición. Por este motivo, la estructura del capítulo será la misma que en los predecesores, es decir, comenzaremos por el módulo de proceso de negocio, luego el módulo de la instancia correspondiente a los atributos, módulo de estados, y función objetivo. Destacar que por motivos de alcance del proyecto, la parte correspondiente a los niveles jerárquicos de conjunto de ejecuciones (Execution Set), y ejecución del proceso (Process Execution), no han podido ser diseñados e implementados. La razón es únicamente por motivo de alcance del proyecto, no obstante, en el siguiente capítulo se presentará un ejemplo real y completo, en el que se incluye la parte de simulación realizada con Excel.

5.1. NIVEL DE MODELO DE PROCESO

En este apartado se explicará en primer lugar la pantalla inicial que presenta la aplicación y que ofrece varias posibilidades, para pasar posteriormente a explicar los distintos formularios de definición del modelo de proceso de negocio. Además, se explicará el sistema de tratamiento de plantillas integrado en la aplicación y que dota a la herramienta de gran rapidez y potencia. Todo ello ayudado de imágenes de los diferentes formularios, con el fin de añadir claridad a las explicaciones. Estos formularios serán presentados en las figuras completados con la definición de un modelo de negocio a modo de ejemplo basado en la reserva de habitaciones de un Hotel.

El primer paso en el diseño de la herramienta es la necesidad de una pantalla inicial, mostrada en la figura 5-1, y que tiene varias finalidades. En primer lugar sirve para familiarizar al usuario con la aplicación RM Modeler 2.0, y conocer el entorno de trabajo, así como una breve introducción que sirve como guía. Estas instrucciones corresponden con el texto que figura en la pantalla inicial y que especifican los pasos a seguir:

- Primero se debería crear un modelo de proceso de negocio, o en caso de tener ya alguno definido en la base de datos abrir el deseado. Una vez tenemos el modelo de proceso se pasa a definir la instancia del proceso, y finalmente tras definir los últimos parámetros para el conjunto de ejecuciones a simular, el programa pasa a

simular la ejecución del proceso. La herramienta guiará al usuario a través del proceso de definición de todas las propiedades requeridas.

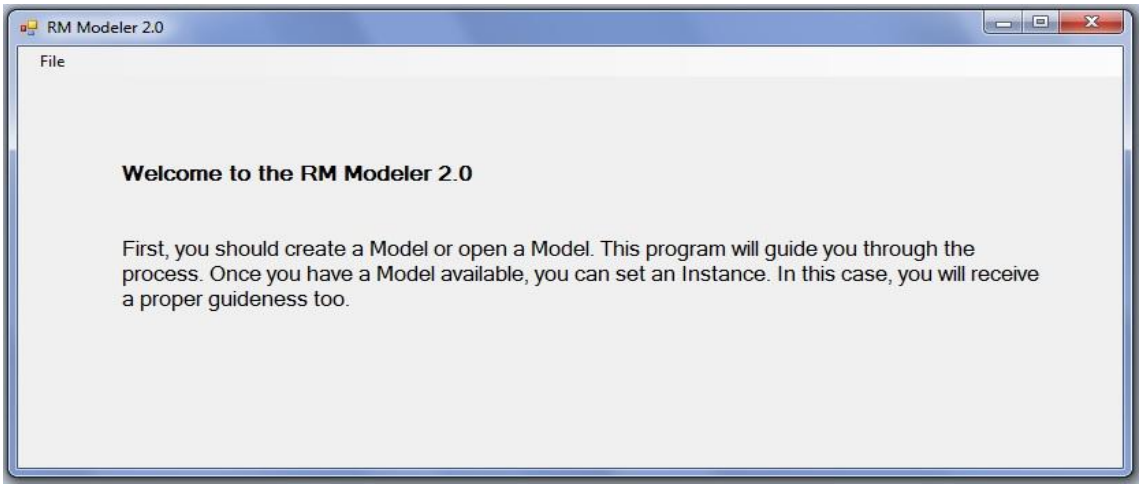


Figura 5-01. Pantalla inicial del RM Modeler 2.0

La segunda utilidad de la pantalla inicial es que cada vez que se guarda un modelo de proceso o una instancia, o se termina la resolución de un problema, se volverá a dicha pantalla inicial. Y en tercer lugar, gracias al menú desplegable mostrado en la figura 5-2, se pueden hacer las siguientes operaciones:

- Se puede comenzar la definición de un nuevo modelo de proceso de negocio a través de la pestaña "New". En este caso, la aplicación pedirá que asignes un nombre al nuevo modelo y una fecha para poder identificarlo más precisamente. Una vez asignado el nombre y fecha, pasaremos al primer formulario.
- Se puede abrir un modelo de proceso de negocio o una instancia de proceso que hayan sido guardados previamente. Si queremos abrir un modelo de proceso ya existente, se abrirá una ventana diálogo mostrando todos los modelos de proceso para poder seleccionar el que se pretende cargar y pasaremos al formulario 1 con los datos cargados. Para el caso de una instancia, primero se seleccionará el modelo de proceso al que pertenece la instancia a cargar, y posteriormente la propia instancia, con lo que pasaríamos al formulario 6 que es donde empieza la definición de la instancia.
- Otra opción es la de borrar (Delete). Esta opción ofrece la posibilidad de borrar tanto un modelo de proceso, con sus respectivas instancias, como alguna plantilla de modelo de proceso que hayamos definido como tal, o una plantilla de instancia que este definida como tal. Además, en caso de querer formatear la base de datos, se ofrece la posibilidad de borrar la base de datos completa (Delete Entire Database).

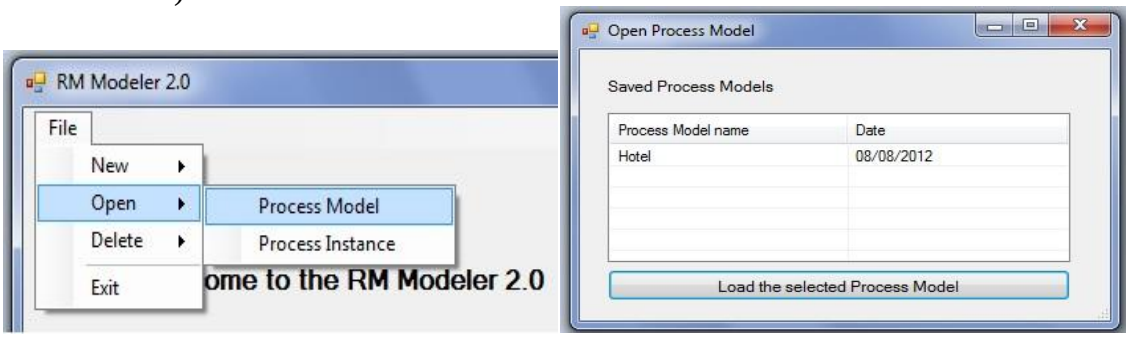


Figura 5-02. Cargar un modelo de proceso desde pantalla inicial

Una vez que el usuario, utilizando la opción "new", elige crear un nuevo modelo de proceso de negocio, se debe introducir un nombre y una fecha identificativos. Si el nombre coincide con alguno de los modelos de negocio ya guardados, aparecerá un mensaje de advertencia de que se debe elegir otro nombre. Ya aceptado el nombre de referencia para el nuevo modelo de proceso de negocio, pasaremos al formulario1, en el que se comienza la definición del modelo de proceso.

FORMULARIO 1 – DEFINICÓN DE ELEMENTOS CUSTOMER SEGMENT, CHANNEL, INFRASTRUCTURE SET, Y VALUE

En la figura 5-3 se muestra el formulario 1 de la aplicación. Este formulario dispone de varias funcionalidades que se activan a través de los diferentes botones y el menú desplegable "File" en la esquina superior izquierda.

Vemos que en la parte superior izquierda aparecerá el nombre del modelo de proceso que hemos designado, en el ejemplo de la figura 5-3 "Hotel". El objetivo principal de este formulario respecto a la definición del modelo de proceso es la definición de los elementos del modelo correspondientes a las entidades Customer Segment (Segmento de cliente), Channel (Canal), Infrastucture (Infraestructura), y Value (Valor). Para ello, mediante los botones "Add", situados a la derecha de cada lista, se pueden añadir un elemento asignándole un nombre. Los botones "Edit" sirven para editar el nombre de cada elemento, por lo que primero se ha se seleccionar el elemento a editar clicando encima del nombre, y posteriormente presionando el botón "Edt". El tercer botón, "Remove", sirve para borrar un elemento ya definido, procediendo de la misma manera que en el caso anterior. Además, debajo de cada lista de elementos se muestra el número de elementos definidos de cada entidad. Mencionar que para que los nombre asignados a cada elemento definido sean aceptados deben cumplir tres condiciones:

- Deben tener más de un carácter.
- El primer carácter no puede ser un espacio en blanco.
- El nombre no puede coincidir con otro elemento de la misma entidad ya definido en el mismo proceso de negocio.

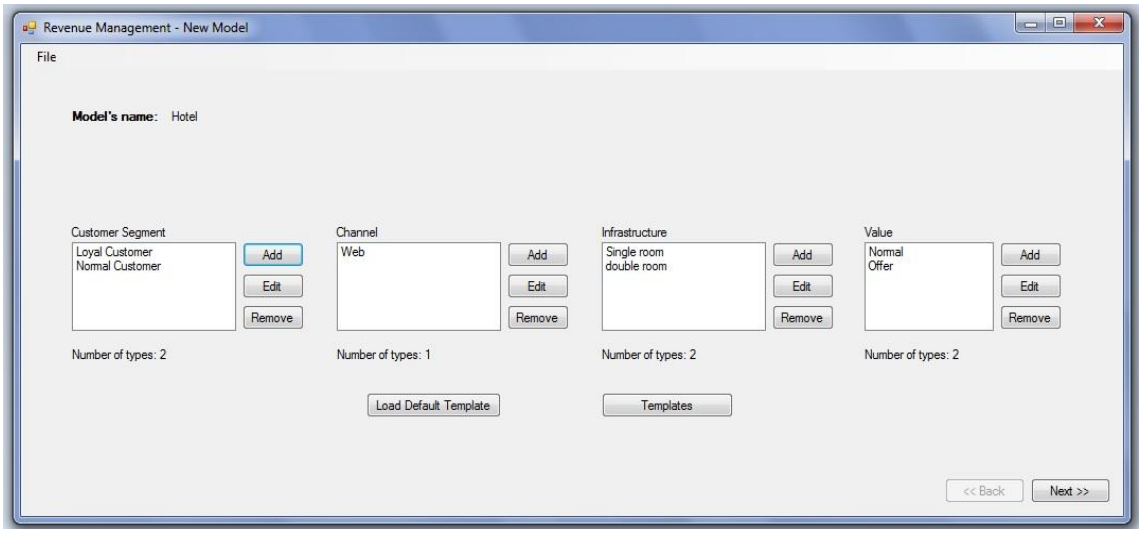


Figura 5-03. Formulario 01 de definición de un modelo de proceso (Process Model)

Por otro lado, tenemos el menú desplegable "File", el cual estará presente en cada uno de los formularios correspondientes a la definición del proceso de negocio y que se explicarán a continuación. Este menú tiene las opciones presentadas en la figura 5-4. Por un lado, podemos salir en cualquier momento a través de la opción "Exit", lo cual cerrará la aplicación. Por otro lado, tenemos la opción de guardar el modelo (Save Process Model) de varias maneras. La primera opción, que corresponde con la seleccionada en la figura 5-4, es guardar los datos como modelo de proceso, lo cual guardará todos los datos como pertenecientes al modelo que está siendo definido. Las dos otras opciones de guardar el modelo están relacionadas con el tratamiento de plantillas que se ha implantado en la aplicación del proyecto y que explicaremos a continuación. Dentro de estas opciones, podemos guardar el modelo como una plantilla (As Process Model Template), o como la plantilla por defecto (As Process Model Default Template).

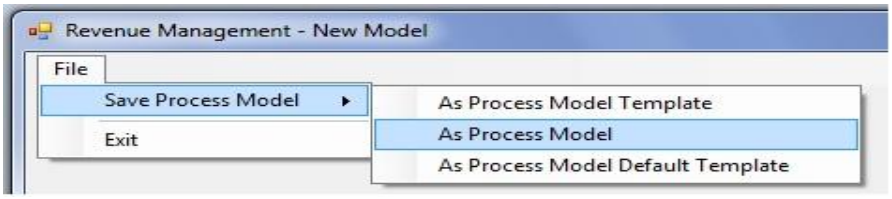


Figura 5-04. Menú desplegable para guardar modelo de proceso

El último aspecto importante a destacar en el formulario 1 son los botones "Load Default Template" y "Templates". Estos dos botones están relacionados con el sistema de tratamiento de plantilla mencionado anteriormente. El primer botón permite cargar el modelo de proceso de negocio guardado como plantilla por defecto, mientras que el segundo botón permitirá cargar cualquiera de las plantillas de modelo de proceso guardadas. Al seleccionar el segundo botón, se mostrará una ventana con la lista de plantillas de modelo de proceso guardadas en la base de datos, tanto el nombre como la fecha de referencia de cada una de ellas, y se seleccionará la deseada.

Una vez terminada la definición de los elementos de cada una de las entidades presentes en este formulario, el usuario debe presionar en el botón "Next" de la esquina inferior derecha si quiere continuar el proceso de definición del modelo de proceso. En este punto, la aplicación comprobará si hay al menos un elemento definido por entidad, y en caso afirmativo se cargará el formulario 2.

TRATAMIENTO DE PLANTILLAS DE MODELO DE PROCESO DE NEGOCIO

Como ya se ha introducido anteriormente, durante el diseño de la aplicación se decidió implantar un tratamiento de plantillas que permitiese al usuario definir nuevos modelos de procesos de negocios basados en modelos ya definidos anteriormente. De esta manera, la herramienta gana en rapidez y aumenta las posibilidades ofrecidas al usuario. Gracias a esta opción, el usuario podrá cargar un modelo guardado como plantillas y modificar o hacer los ajustes deseados para confeccionar un nuevo modelo.

En caso de querer estudiar modelos de negocio similares, gracias a esta funcionalidad no es necesario definir desde cero cada uno de los modelos, bastará con definir uno, guardarlo como plantilla, y emplearlo como base para la definición del modelo similar.

El sistema de tratamiento de plantillas consta de dos tipos de plantilla, la plantilla denominada por defecto (Default), y las plantillas normales. El motivo de diferenciar una plantilla y guardarla como plantilla por defecto reside en ofrecer al usuario mayor rapidez y comodidad en el manejo de plantillas. Es decir, si el usuario maneja muy a menudo una plantilla, o requiere del uso más frecuentemente que el resto, parece lógico tener un acceso directo a dicha plantilla, y por ello en lugar de buscar entre las plantillas existentes se emplaza como plantilla por defecto. Para aplicar esta filosofía, en el formulario 1 hemos explicado el botón "Load Default Template", que cargará automáticamente la plantilla por defecto. Al guardar una plantilla por defecto, la plantilla que había anteriormente en este lugar pasará a ser una plantilla normal, mientras que la nueva plantilla ocupará el lugar Default.

Las plantillas serán almacenadas en la base de datos con identificativos negativos, es decir, con ID_ProcessModel negativos, reservando el identificativo -1 para la plantilla por defecto y el resto para las plantillas normales. Se trata de la técnica elegida para poder diferenciar con las mismas tablas en la base de datos, entre los modelos de proceso y las plantillas de modelos de proceso.

Al igual que los propios modelos de proceso guardados en la base de datos, las plantillas se identifican por un nombre y una fecha, de manera que el nombre será único para cada plantilla. En cuanto a la plantilla por defecto, la aplicación incorpora una plantilla ya definida por defecto, llamada "Default" y que puede servir al usuario en sus comienzos con el uso de la aplicación. Esta plantilla se implanto para servir como modelo base o como ejemplo de definición de un proceso de negocio, que permitiera al usuario familiarizarse con la herramienta desde el comienzo y de forma rápida.

Como hemos comentado anteriormente, en cada uno de los formularios de definición de proceso de negocio se ofrece la posibilidad de guardar el modelo como plantilla normal o plantilla por defecto. Esto de puede hacer a través del menú desplegable "File"", situados en la esquina superior izquierda de cada uno de los formularios y mostrados en la figura 5-4. Se ofrece dicha posibilidad en cada formulario pues la necesidad de guardar un modelo como plantilla puede surgir en cualquiera de los pasos en el proceso de definición.

FORMULARIOS 2 Y 3 – ELECCIÓN DE LOS INFRASTRUCTURE ACCESS

Los formularios dos y tres están orientados a la definición de los elementos correspondientes a la entidad Infrastructure Access (Acceso a infraestructuras) que el usuario desea habilitar.

Recordar que la entidad Infrastructure Access es consecuencia de la combinación de la terna cliente, canal, e infraestructura. Por ello, el usuario deberá elegir entre todas las posibles combinaciones de los elementos correspondientes a las entidades normbradas (Terna), con el fin de que el modelo contemple esos accesos a infraestructuras. En el formulario 2 se muestran todas estas posibles combinaciones, entre las que el usuario debe seleccionar las que desee incluir en la definición de los elementos de la entidad Acceso a infraestructura. En la figura 5-5 se muestra el formulario 2, con el ejemplo de reservas de habitaciones de hotel que estamos empleando.

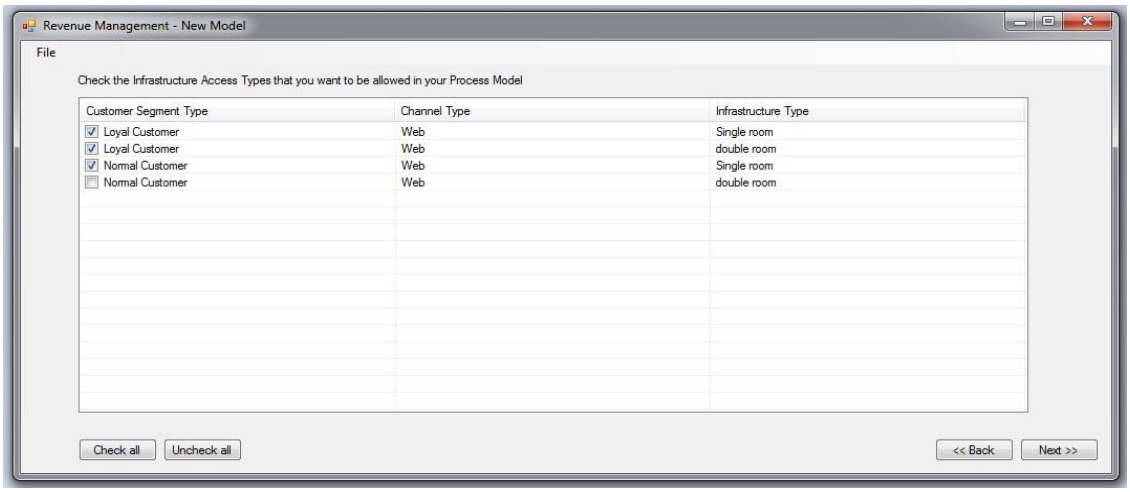


Figura 5-05. Formulario 02 de definición de un modelo de proceso (Process Model)

Como vemos en la figura 5-5, en este ejemplo habría cuatro posibles combinaciones, es decir, cuatro posibles elementos de la entidad Infrastructure Access. El usuario debe seleccionar los deseados cliqueando en la casilla al lado del nombre correspondiente. En un modelo nuevo, por defecto aparecerán las casillas no seleccionadas; pero en el caso de un modelo cargado que ya había sido definido previamente, aparecerán seleccionadas las casillas que lo estuviesen cuando se guardó el modelo.

Los dos botones situados en la parte inferior izquierda, están orientados a aportar rapidez al proceso de definición. El botón "Check all" (Selecionar todos) sirve para seleccionar automáticamente todas las opciones, mientras que el botón "Uncheck all" (Deseleccionar todos) sirve para deseleccionar todas las combinaciones. Como siempre, en la parte superior izquierda está el menú desplegable "File", ya explicado. Por último, en caso de querer volver al formulario anterior de definición del modelo para realizar cambios o consultas, se presenta el botón "Back" (Atrás). Para continuar al siguiente formulario se debe seleccionar el botón "Next" (Siguiente), momento en el que la herramienta comprueba si al menos una combinación ha sido seleccionada y tras guardar las propiedades se accede al formulario 3 (Figura 5-6).

El formulario 3, mostrado en la figura 5-6, se emplea en forma de resumen para mostrar al usuario las combinaciones elegidas en el formulario 2, es decir, los elementos de la entidad Infrastructure Access que forman parte de la definición del proceso. Los botones Next y Back tienen las mismas funciones que en el caso anterior, por lo que al seleccionar el botón Next, la aplicación nos mostrará el formulario 4.

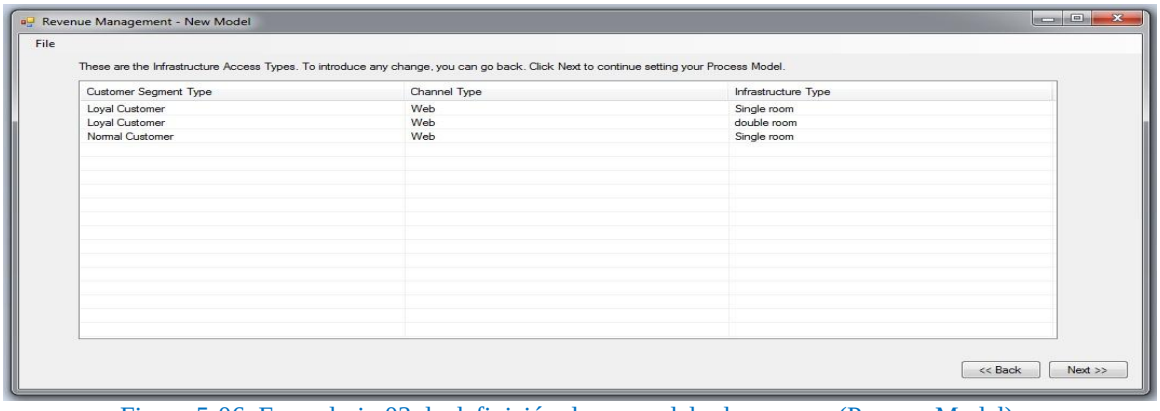


Figura 5-06. Formulario 03 de definición de un modelo de proceso (Process Model)

FORMULARIO 4 – DEFINICIÓN DE LOS ELEMENTOS DE LA ENTIDAD TIME

El formulario 4 corresponde con la definición de los elementos asociados a la entidad Time (Tiempo) y las restricciones temporales entre ellos (Constraints).

En la figura 5-7 se muestra el formulario 4. En la parte izquierda del formulario se presenta una lista con los elementos de tiempo definidos. Siempre habrá mínimo dos elementos, marcados como T.Start (Tiempo inicio) y T.End (Tiempo final), y que representan el inicio y el fin de la ejecución respectivamente. Con los botones situados a la derecha de la lista se pueden añadir (Add) o borrar (Remove) elementos definidos, teniendo en cuenta que ni T.Start ni T.end se pueden borrar. Los nombres de cada elemento se asignan de manera automática por la aplicación, siguiendo la nomenclatura T.1, T.2, T.3, etc.

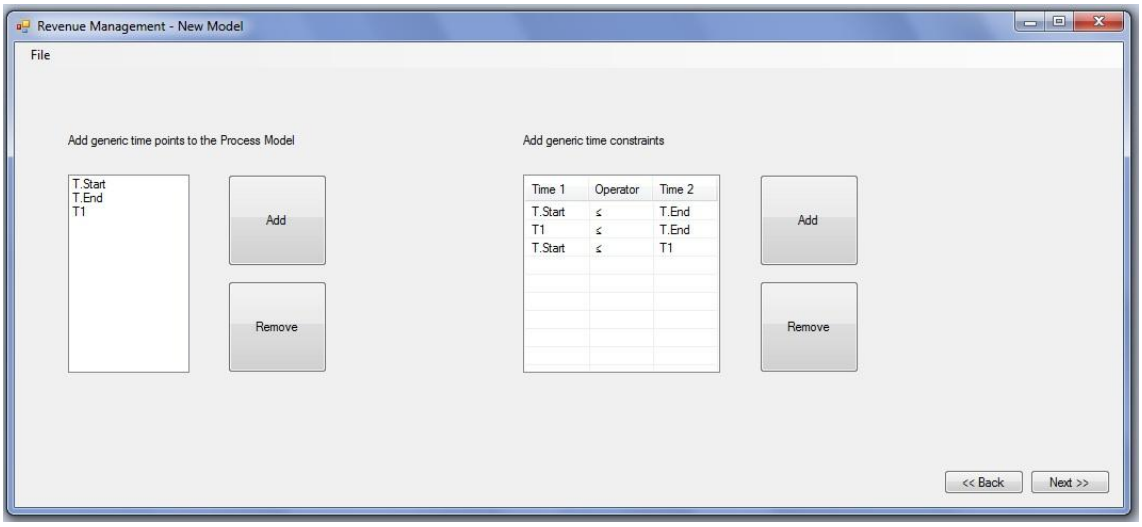


Figura 5-07. Formulario 04 de definición de un modelo de proceso (Process Model)

Por otro lado, la parte derecha del formulario está relacionada con la definición de relaciones de restricción entre los elementos de tiempo que figuran en la parte izquierda y que han sido ya definidos. Como vemos, la tabla de la derecha consta de dos columnas referidas a los tiempos que se relacionan, y la columna central que expresa la relación entre ambas. Cabe destacar que cualquier tiempo definido será mayor o igual a T.Start y al mismo tiempo menor o igual que T.End. Por ello, cada vez que se añada un elemento de tiempo nuevo, estas restricciones se expresaran automáticamente en la ventana de restricciones y serán guardadas, no permitiendo al usuario borrarlas.

Los dos botones de la parte derecha permiten añadir (Add) y borrar (Remove) restricciones temporales. Hay que tener en cuenta que cuando se borra un elemento de tiempo, todas las restricciones temporales relacionadas con dicho elementos serán borradas. Si se selecciona el botón "Add", se presentará el cuadro de diálogo de la figura 5-8. En este cuadro, se debe elegir el elemento de tiempo del punto 1, a través de la casilla desplegable que sirve de filtro. También se debe seleccionar el elemento de tiempo del punto 2, y el operador que refleje la relación entre ambos. Como operadores de relación sólo se pueden emplear "=", o bien "≤", y la aplicación no permitirá crear una restricción entre elementos que están ligados por una restricción ya existente. Por último, destacar que cuando se define una relación del tipo menor o igual, el índice del elemento de tiempo que es menor no puede ser superior al otro elemento. Por ejemplo,

T1 puede ser igual o menor que T2, pero en ningún caso T2 puede ser definido como menor o igual que T1. Sin embargo, T1 y T2 sí pueden ser iguales.

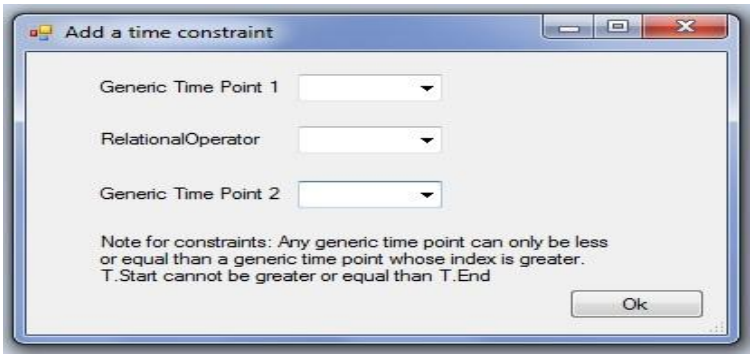


Figura 5-08. Ventana de diálogo de definición de restricciones temporales

Al igual que en el formulario anterior, el formulario 4 de la figura 5-7 consta tanto del menú desplegable "File", como de los botones Next y Back. Cuando el usuario pulsa el botón Next, antes de pasar al formulario 5, la aplicación genera dos matrices que sirven para comprobar que no existan solapes entre los elementos de la entidad Allocation (Asignación) que cuenten con idénticos segmento de cliente, canal, infraestructura, y valor, ya que no es posible ofrecer al mismo tiempo el mismo tipo de infraestructura, por el mismo canal, al mismo segmento de cliente y con distinto valor. Por ejemplo, no se puede ofrecer al mismo tiempo una habitación simple, para ser reservada por internet, por clientes fieles, y con precio normal y de oferta a la vez, sería un precio u otro.

FORMULARIO 5 – DEFINICIÓN DE LOS ELEMENTOS DE LA ENTIDAD ALLOCATION

El formulario 5 tiene la finalidad de definir los distintos elementos pertenecientes a la entidad Allocation (Asignación). En la figura 5-9 se muestra el formulario 5 completado con el ejemplo de reserva de habitaciones de un Hotel.

La lista de la parte superior del formulario consta de los diferentes elementos de la entidad Infrastructure Acces (Acceso a infraestructura) definidos en el formulario 2 por el usuario. A partir de esta lista, seleccionando primero el elemento infrastructure acces a partir del cual se quiere definir un nuevo elemento allocation, y posteriormente presionando el botón "Add Allocation Element" situado a la derecha de la lista, nos aparecerá el cuadro de diálogo de la figura 5-10. En este cuadro, el usuario debe seleccionar el espacio temporal (Start y End) que caracterizará al elemento allocation, y el valor asociado; y tras presionar en el botón Ok el nuevo elemento allocation será mostrado en la lista inferior del formulario, siempre que el tiempo Start sea inferior o igual al tiempo End.

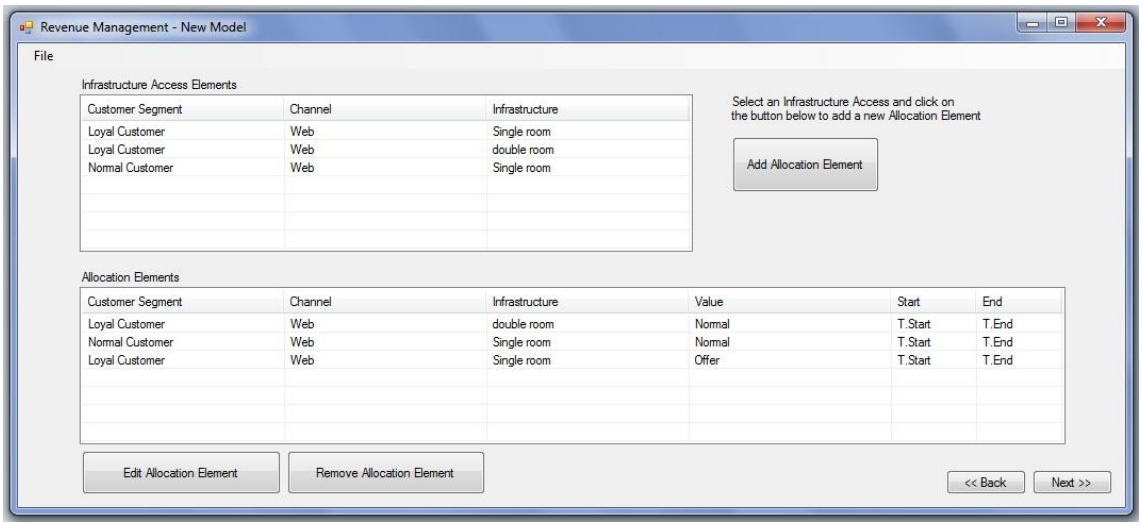


Figura 5-09. Formulario 05 de definición del modelo de proceso (Process Model)

En la lista inferior del formulario 5 por tanto, se muestran los elementos allocation que son definidos. Estos elementos pueden ser borrados, si selecciono el elemento a borrar y posteriormente el botón "Remove Allocation Element", o pueden ser editados, seleccionando el elemento de la lista a editar y posteriormente en el botón "Edit Allocation Element". Al escoger la opción editar, se mostrará el mismo cuadro de diálogo que al añadir un nuevo elemento, es decir, el cuadro de la figura 5-10, pudiéndose editar el espacio temporal y el valor, pero no la infraestructura de acceso.

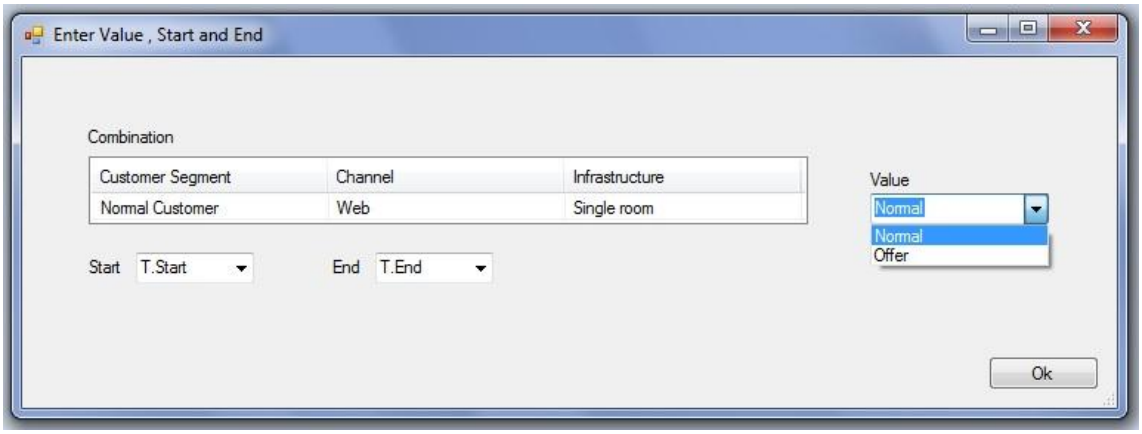


Figura 5-10. Cuadro de diálogo de configuración de un elemento Allocation

Finalmente, mediante los botones Next y Back, situados en la parte inferior derecha del formulario 5, podemos volver al formulario anterior o pasar al siguiente paso en el proceso de definición respectivamente. La aplicación contiene una subrutina que se ejecutará tanto si pulsamos Next, como si pulsamos Back, o si guardamos el modelo a través del menú desplegable "File", por la que antes de guardar los elementos allocation se comprueba una condición necesaria. Esta condición consiste en que los allocation con el mismo cliente, canal, e infraestructura, no pueden solaparse en un mismo intervalo temporal. Si no se cumple esta condición, la aplicación avisará de qué elementos allocation de la lista están incumpliendo dicha norma.

Si el usuario selecciona el botón Next, al menos un elemento allocation ha sido definido, y se cumple la norma de no solapamiento explicada, la aplicación mostrará la

ventana de la figura 5-11 de finalización del proceso de definición del modelo de proceso de negocio.

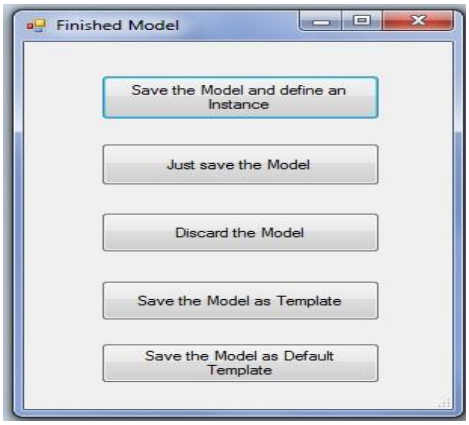


Figura 5-11. Cuadro de diálogo de finalización de definición del modelo de proceso

En la ventana de la figura 5-11 se ofrecen varias opciones:

- Just save the model: Mediante esta opción el modelo definido será guardado y se volverá a la pantalla inicial de la aplicación.
- Discard the Model: Se trata de borrar el modelo que ha sido definido. De nuevo, la aplicación volverá a la pantalla inicial.
- Save the Model as Template: Guardar el modelo como plantilla.
- Save the Model as Default Template: Guardar el modelo como plantilla por defecto.
- Save the Model and define an Instance: Guardar el modelo y definir una instancia de proceso. Con esta opción se pasará al proceso de definición de una instancia de proceso, con lo que pasaremos al formulario 6 y comenzaremos a definir el módulo de atributos.

5.2. NIVEL DE INSTANCIA DE PROCESO

En el apartado anterior se ha explicado el módulo de la aplicación correspondiente a la definición del modelo de proceso de negocio a través de los cinco primeros formularios. En este apartado se pasa al nivel de instancia de proceso, por lo que el objetivo es describir el funcionamiento de la aplicación a través de la definición de los módulos de la instancia de proceso. Para ello, al igual que en capítulos anteriores, empezaremos por el módulo de atributos, luego pasaremos al módulo de estados, y acabaremos con el módulo de la función objetivo.

En primer lugar, y como se ha explicado ya tanto en el capítulo de modelos conceptuales como en el de modelo de datos, una instancia de un modelo de proceso en "hermana" del modelo de proceso, pues ambos son subtipos de modelo de asignación de infraestructuras, aunque se diferencian en que la instancia es más concreta. Por este motivo, los elementos definidos mediante la aplicación en correspondientes al modelo de proceso, serán también elementos de la instancia de proceso, aunque al nivel de instancia incorporarán más propiedades que serán las que definamos a continuación y empezando por los atributos.

5.2.1 MÓDULO DE DEFINICIÓN DE ATRIBUTOS

En este apartado se describen los distintos formularios y algunos cuadros de diálogos enfocados a la definición de los atributos y su caracterización.

Una vez se ha seleccionado la opción de definir una nueva instancia de proceso, la aplicación requerida que el usuario identifique la nueva instancia con un nombre y una fecha. Si el nombre introducido no está ya asignado a ninguna otra instancia, la aplicación nos mostrará el formulario 6 mostrados en la figura 5-12.

FORMULARIO 6 – DEFINICIÓN DE LOS ATRIBUTOS DE LA INSTANCIA

El formulario 6 sirve para dar soporte al usuario en la definición de los atributos de la instancia. En la parte superior izquierda se presenta tanto el nombre del modelo de proceso como el nombre de la instancia en curso, con el fin de exponer al usuario una clara visión de a quién están asociados los atributos.

Como podemos observar en la figura 5-12, existen siete listas distintas, cada una de ellas está asociada a una entidad del modelo distinta, ya que como explicamos en los modelos conceptuales cada definición de atributo va vinculado a una entidad del modelo. Es decir, hay una lista para los atributos asociados a la entidad segmento de cliente, otra para canal, y así con las siete entidades. Cada lista dispone a su derecha de tres botones asociados, denominados Add (Añadir), Edit (Editar), y Remove (Borra):

- Remove: Sirve para eliminar el atributo seleccionado previamente de la lista.
- Add: Permite al usuario añadir nuevos atributos a la lista correspondiente. Al pulsar este botón, se abrirá el cuadro de diálogo de la figura 5-13, donde se debe especificar el nombre y el tipo de atributo a añadir. El nombre no puede coincidir con el de ningún otro atributo, y el tipo debe ser elegido entre las opciones que aparecen en la lista desplegable de la figura 5-13.
- Edit: Permite al usuario modificar las características previamente definidas del atributo que ha sido seleccionado. Emplea el mismo cuadro de diálogo que para añadir un atributo, es decir, el de la figura 5-13. Al igual que al añadir un atributo, si modifico el nombre éste no podrá coincidir con ningún otro nombre de atributo.

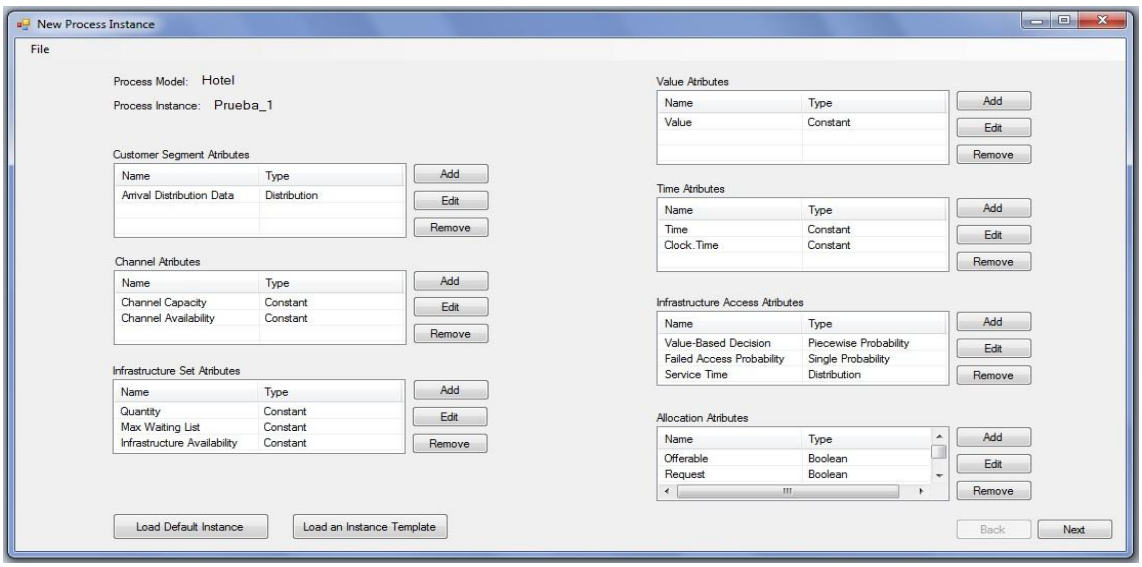


Figura 5-12. Formulario 06, primero del módulo de definición de atributos

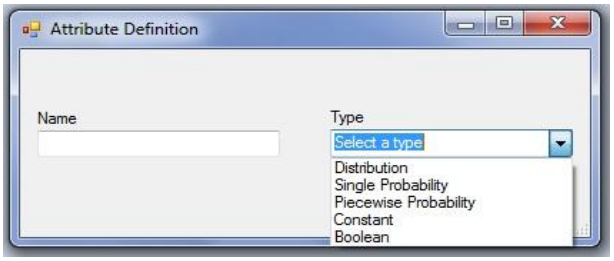


Figura 5-13. Cuadro de diálogo de adición y edición de atributos

Al igual que sucedía en todos los formularios correspondientes a la definición de un modelo de proceso, todos los formularios pertenecientes a la definición de una instancia irán acompañados en la esquina superior izquierda del menú desplegable "File" presentado en la figura 5-14.

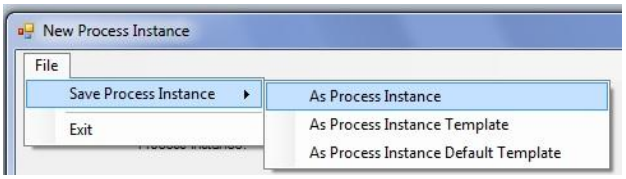


Figura 5-14. Menú desplegable "File" para guardar una instancia de proceso

En este caso, las opciones presentadas son la de salida (Exit), por lo que la aplicación se cerrará, o las tres opciones de guardar la instancia. Podemos guardas la instancia como una instancia normal (As Process Instance), como una plantilla normal de instancia (As Process Instance Template), o como una plantilla de instancia por defecto (As Process Instance Default Template). De nuevo, estas opciones permiten al usuario guardar la instancia en cualquier momento y formulario.

Por último, los dos botones de la esquina inferior izquierda del formulario 6 de la figura 5-12, están relacionados con el sistema de tratamiento de plantillas de una instancia. El botón "Load Default Instance" permite cargar la instancia guardada por defecto, mientras que el botón "Load an instance template" dará lugar a un cuadro de diálogo que muestra todas las plantillas de instancias guardadas, para poder seleccionar la deseada y cargarla. A continuación se detalla el tratamiento de plantillas de las instancias. El botón Next lleva a la siguiente paso, que es el formulario 7.

TRATAMIENTO DE PLANTILLAS DE INSTANCIA DE PROCESO DE NEGOCIO

Al igual que en el nivel de modelo de proceso, también hay un sistema de plantillas implantado en el nivel de instancia de proceso. El propósito de este sistema es también ofrecer al usuario la posibilidad de usar instancias guardadas como plantillas para crear una nueva instancia a partir de ellas. Este sistema ofrece mayor rapidez y es una herramienta muy útil y versátil que facilita la definición de instancias.

Si el usuario desea basar la nueva instancia en una plantilla ya guardada, puede cargar la plantilla de la instancia deseada, opción presente en el formulario 6 y que se ha explicado previamente, de manera que ésta se adaptará al modelo de proceso actual. Las plantillas de instancias posibilitan la definición, de una forma rápida, de instancias muy similares pero que difieren en pequeños aspectos.

Una vez más, habrá dos tipos de plantillas de instancia, la plantilla de instancia por defecto (Default) y las plantillas de instancia normales.

- La *plantilla de instancia por defecto* (Instance Default Template) será identificada en la base de datos por los atributos: ID_ProcessModel= 0, ID_ProcessInstance= -1. Y está orientada a la plantilla de la instancia que se quiera usar más frecuentemente. La aplicación presenta una plantilla de instancia por defecto inicial para ayudar al usuario a familiarizarse con la herramienta y para servir también como modelo, pero ésta puede ser sustituida en cualquier momento.
- Las *plantillas normales de instancias*, se identifican con ID_ProcessModel=0, y para ID_ProcessInstances números negativos distintos del -1, reservado para la plantilla por defecto. Técnica para implementar en la misma base de datos las plantillas y las instancias normales. Se pueden guardar tantas plantillas como el usuario requiera, siempre identificadas por un nombre y una fecha.

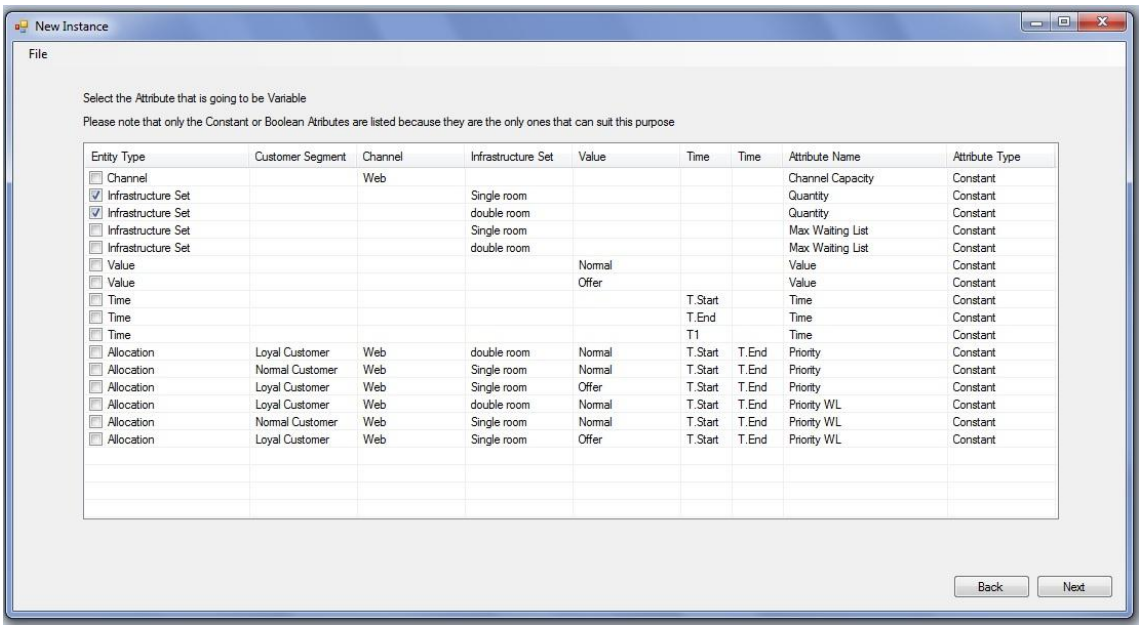
Como se ha explicado, en cualquier formulario de definición de la instancia ésta se puede guardar tanto como plantilla normal como plantilla por defecto, a través del menú desplegable "File" situado en la esquina superior izquierda.

FORMULARIO 7 – DEFINICIÓN DE LAS VARIABLES DEL PROBLEMA

En el formulario 7, mostrado en la figura 5-15, la finalidad es definir los atributos que van a formar las variables del problema de asignación de infraestructuras, y que por lo tanto se pretenden optimizar tras la simulación del proceso.

El formulario consta básicamente de la lista de atributos que son candidatos a ser definidos como variables del problema. Destacar que sólo los atributos de tipo constante (Constant) y de tipo booleano (Boolean) pueden ser variables de problema, ya que los otros dos tipos de atributos, de probabilidad y de distribución, tienen componente aleatoria y la optimización de las variables no estaría garantizado. Junto a cada nombre de atributo (Attribute Name) se presentan las características del elemento del modelo al que está asociado así como la entidad (primera columna). Para seleccionar un atributo como variable, el usuario dispone de unas casillas a la izquierda de cada uno que se pueden cliquear.

Mediante el botón "Back" de la esquina inferior derecha se vuelve al formulario anterior. En caso de que el usuario vuelva al formulario anterior y cambie las características de las variables del problema, éstas modificaciones afectarían como es lógico al formulario actual. Finalmente, para pasar al formulario 8 se debe seleccionar el botón "Next", y la aplicación avanzará al siguiente formulario siempre que se haya definido al menos un atributo como variable.



Entity Type	Customer Segment	Channel	Infrastructure Set	Value	Time	Time	Attribute Name	Attribute Type
Channel		Web					Channel Capacity	Constant
Infrastructure Set			Single room				Quantity	Constant
Infrastructure Set			double room				Quantity	Constant
Infrastructure Set			Single room				Max Waiting List	Constant
Infrastructure Set			double room				Max Waiting List	Constant
Value				Normal			Value	Constant
Value				Offer			Value	Constant
Time					T.Start		Time	Constant
Time					T.End		Time	Constant
Time					T1		Time	Constant
Allocation	Loyal Customer	Web	double room	Normal	T.Start	T.End	Priority	Constant
Allocation	Normal Customer	Web	Single room	Normal	T.Start	T.End	Priority	Constant
Allocation	Loyal Customer	Web	Single room	Offer	T.Start	T.End	Priority	Constant
Allocation	Loyal Customer	Web	double room	Normal	T.Start	T.End	Priority WL	Constant
Allocation	Normal Customer	Web	Single room	Normal	T.Start	T.End	Priority WL	Constant
Allocation	Loyal Customer	Web	Single room	Offer	T.Start	T.End	Priority WL	Constant

Figura 5-15. Formulario 07, definición de variables de la instancia

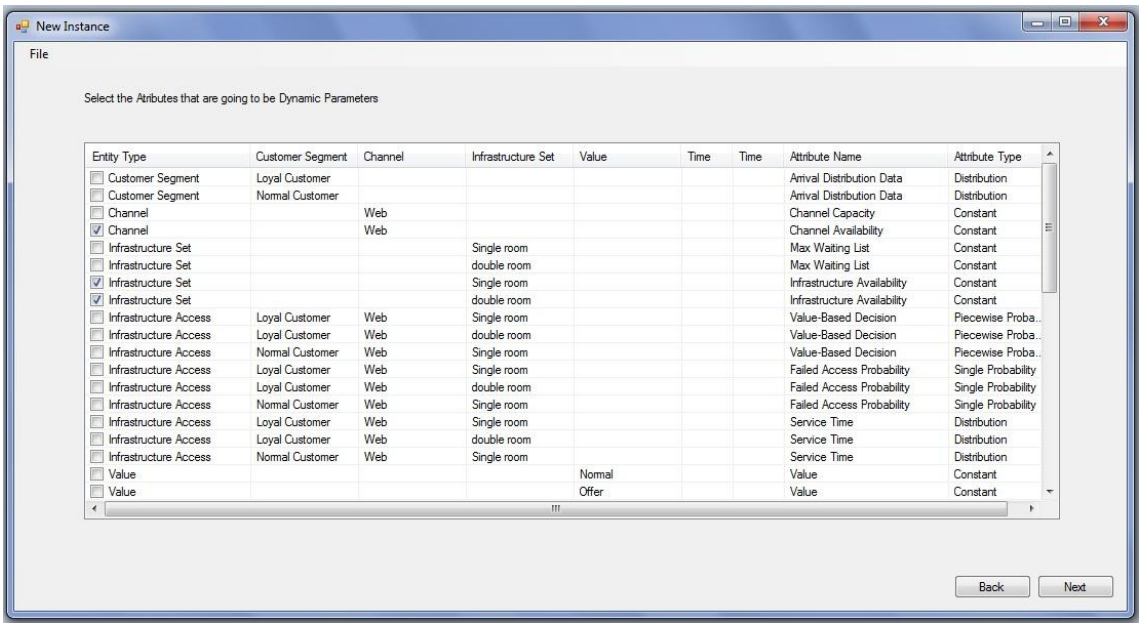
FORMULARIO 8 – DEFINICIÓN DE LOS ATRIBUTOS DINÁMICOS

En el formulario 8, tercero en la definición de la instancia, se definen todos aquellos atributos que van a ser dinámicos. En la figura 5-16 se muestra el aspecto del formulario relleno con el ejemplo de reserva de habitaciones de un hotel.

La estructura del formulario es igual que el anterior, pero en este caso la lista de atributos incluye todos los atributos, de todos los tipos, excepto los que fueron definidos como variables del problema en el formulario anterior. Cada atributo se identifica por su nombre (Attribute Name), y por el elemento del modelo al que va asociado y que se refleja en las columnas pertinentes, además la primera columna contiene la entidad del modelo asociada. Las casillas de la parte izquierda de la lista permiten seleccionar los atributos que el usuario quiera declarar como dinámicos.

Tanto los atributos de evento dinámicos como los atributos dinámicos de ejecución constante deben ser marcados en este formulario, y más adelante se diferenciará entre los dos tipos de atributos dinámicos. Atributos como “Infrastructure Availability” o “Channel Capacity”, ya se explicó en la parte teórica que son esenciales para cualquier instancia y siempre serán dinámicos por exigencias del modelo conceptual, con lo que aunque se deseleccionen en la base de datos permanecerán dinámicos. Por este motivo, al cargar el formulario aparecerán seleccionados por defecto. Los atributos booleanos también aparecerán seleccionados por defecto, aunque se pueden cambiar.

Por último, al igual que sucede en formularios anteriores, cada vez que se vuelva a acceder a este formulario se cargarán las opciones guardadas en la base de datos, por lo que al volver a este formulario desde formularios sucesores las opciones que se presenten marcadas serán las que el usuario definió.



Entity Type	Customer Segment	Channel	Infrastructure Set	Value	Time	Time	Attribute Name	Attribute Type
Customer Segment	Loyal Customer						Arrival Distribution Data	Distribution
Customer Segment	Normal Customer						Arrival Distribution Data	Distribution
Channel		Web					Channel Capacity	Constant
Channel		Web					Channel Availability	Constant
Infrastructure Set			Single room				Max Waiting List	Constant
Infrastructure Set			double room				Max Waiting List	Constant
Infrastructure Set			Single room				Infrastructure Availability	Constant
Infrastructure Set			double room				Infrastructure Availability	Constant
Infrastructure Access	Loyal Customer	Web	Single room				Value-Based Decision	Piecewise Proba..
Infrastructure Access	Loyal Customer	Web	double room				Value-Based Decision	Piecewise Proba..
Infrastructure Access	Normal Customer	Web	Single room				Value-Based Decision	Piecewise Proba..
Infrastructure Access	Loyal Customer	Web	Single room				Failed Access Probability	Single Probability
Infrastructure Access	Loyal Customer	Web	double room				Failed Access Probability	Single Probability
Infrastructure Access	Normal Customer	Web	Single room				Failed Access Probability	Single Probability
Infrastructure Access	Loyal Customer	Web	Single room				Service Time	Distribution
Infrastructure Access	Loyal Customer	Web	double room				Service Time	Distribution
Infrastructure Access	Normal Customer	Web	Single room				Service Time	Distribution
Value				Normal			Value	Constant
Value				Offer			Value	Constant

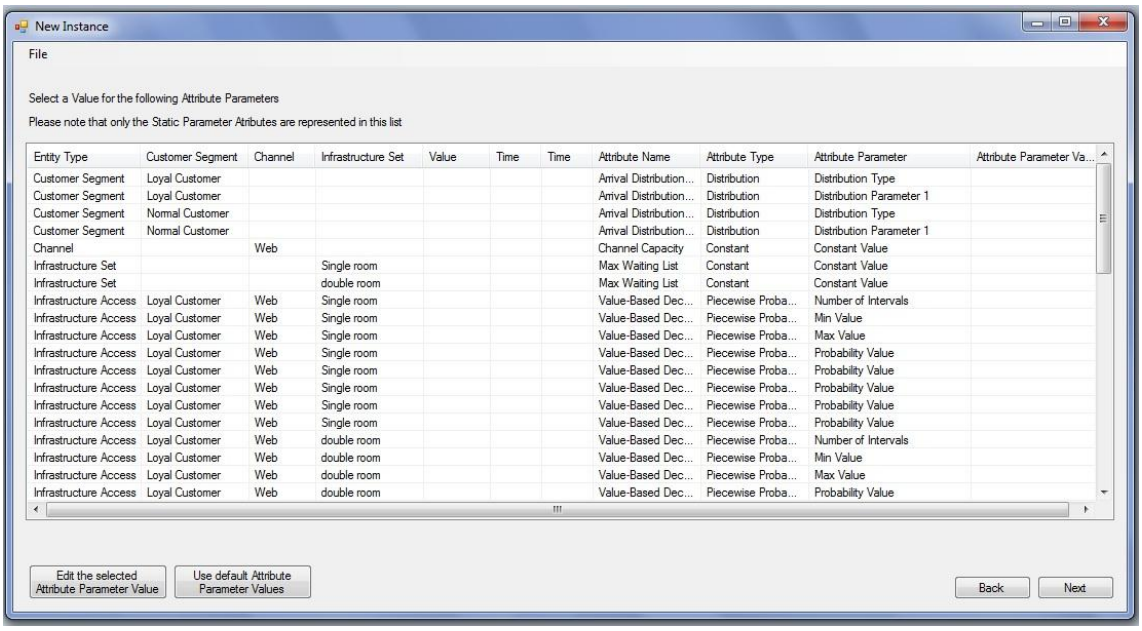
Figura 5-16. Formulario 08, selección de los atributos dinámicos de la instancia

Con el botón “Back” se puede volver al formulario anterior, y si se quiere pasar al siguiente formulario el botón “Next” debe ser seleccionado.

FORMULARIO 9 – DEFINICIÓN DE LOS PARÁMETROS DE LOS ATRIBUTOS ESTÁTICOS

En el formulario 9 se asignan valores a los parámetros de atributo correspondientes a los diferentes atributos de tipo estático. Los valores asignados en este formulario serán iguales para todos los conjuntos de ejecuciones derivados de la instancia, y por lo tanto para todas las ejecuciones.

La figura 5-17 contiene el formulario 9, compuesto por la lista de parámetros de atributo correspondientes a todos los atributos estáticos de la instancia. Cada atributo tendrá un número de parámetros (filas en la lista) en función del tipo de atributo que sea. Así por ejemplo, como se explicó en el módulo de atributos de los modelos conceptuales, un atributo tipo distribution (Distributción) tendrá dos parámetros de atributo asociados, uno para definir el tipo de distribución y otro para definir la media. En la implementación de la aplicación se decidió permitir sólo distribuciones exponenciales, por ser las más comunes en el tipo de problemas en estudio.



Entity Type	Customer Segment	Channel	Infrastructure Set	Value	Time	Time	Attribute Name	Attribute Type	Attribute Parameter	Attribute Parameter Va...
Customer Segment	Loyal Customer						Arrival Distribution...	Distribution	Distribution Type	
Customer Segment	Loyal Customer						Arrival Distribution...	Distribution	Distribution Parameter 1	
Customer Segment	Normal Customer						Arrival Distribution...	Distribution	Distribution Type	
Customer Segment	Normal Customer						Arrival Distribution...	Distribution	Distribution Parameter 1	
Channel		Web					Channel Capacity	Constant	Constant Value	
Infrastructure Set			Single room				Max Waiting List	Constant	Constant Value	
Infrastructure Set			double room				Max Waiting List	Constant	Constant Value	
Infrastructure Access	Loyal Customer	Web	Single room				Value-Based Dec...	Piecewise Proba...	Number of Intervals	
Infrastructure Access	Loyal Customer	Web	Single room				Value-Based Dec...	Piecewise Proba...	Min Value	
Infrastructure Access	Loyal Customer	Web	Single room				Value-Based Dec...	Piecewise Proba...	Max Value	
Infrastructure Access	Loyal Customer	Web	Single room				Value-Based Dec...	Piecewise Proba...	Probability Value	
Infrastructure Access	Loyal Customer	Web	Single room				Value-Based Dec...	Piecewise Proba...	Probability Value	
Infrastructure Access	Loyal Customer	Web	Single room				Value-Based Dec...	Piecewise Proba...	Probability Value	
Infrastructure Access	Loyal Customer	Web	Single room				Value-Based Dec...	Piecewise Proba...	Probability Value	
Infrastructure Access	Loyal Customer	Web	Single room				Value-Based Dec...	Piecewise Proba...	Probability Value	
Infrastructure Access	Loyal Customer	Web	double room				Value-Based Dec...	Piecewise Proba...	Number of Intervals	
Infrastructure Access	Loyal Customer	Web	double room				Value-Based Dec...	Piecewise Proba...	Min Value	
Infrastructure Access	Loyal Customer	Web	double room				Value-Based Dec...	Piecewise Proba...	Max Value	
Infrastructure Access	Loyal Customer	Web	double room				Value-Based Dec...	Piecewise Proba...	Probability Value	

Figura 5-17. Formulario 09, definición de valores de los parámetros de atributos estáticos

Cada línea representa un parámetro (Attribute Parameter) asociado a un atributo concreto (Attribute Name) de un elemento del modelo. Tanto el elemento del modelo como la entidad asociada a cada parámetro de atributo vienen expresados en las columnas de la lista. La columna vacía (Attribute Parameter Value) contendrá los valores asignados cuando sean definidos por el usuario.

Para asignar valores a los parámetros, el formulario dispone de dos botones en la esquina inferior izquierda con funciones diferentes. Por un lado, el botón "Edit the selected Attribute Parameter Value" sirve para editar el valor del parámetro de atributo seleccionado previamente a través del cuadro de diálogo mostrado en la figura 5-18. En el ejemplo de la figura se edita un parámetro de probabilidad, pero el cuadro es sensible al tipo de dato y requerirá un valor dependiendo del tipo de parámetro seleccionado. Por otro lado tenemos el botón "Use Default Attribute Parameter Value". Seleccionando este botón, la aplicación contiene una subrutina que asignará a todos los parámetros de atributo un valor por defecto dependiendo del tipo que sean y, en algunos casos, del nombre del atributo. El objetivo de este segundo botón es aumentar la rapidez de definición de una instancia con valores estándar y que posteriormente, el usuario pueda realizar los ajustes necesarios mediante el botón de editar. Por ello, aumenta la rapidez y mantiene la flexibilidad.

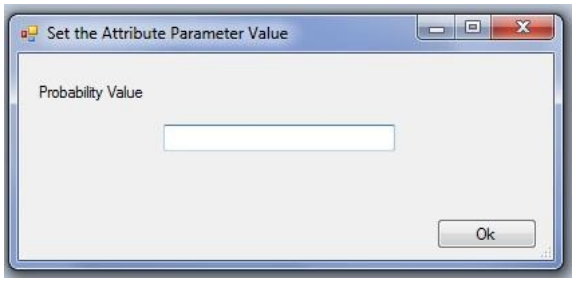


Figura 5-18. Cuadro de diálogo de edición de un parámetro de atributo de tipo probabilidad

Como siempre, mediante el botón "Back" podemos volver al anterior formulario y con el botón "Next" solicitaremos pasar al siguiente formulario. Antes de pasar al

siguiente formulario, la aplicación comprueba que todos los parámetros de atributo han recibido un valor.

Tras haber asignado valor a todos los parámetros de atributo t pulsar el botón "Next", la aplicación nos mostrará el formulario 10 perteneciente al módulo de estados. Por motivos que se exponen a continuación, el siguiente formulario perteneciente a la definición de atributos en la instancia (Módulo de atributos) es el formulario 14. Por lo que para continuar en el módulo de atributos pasamos a explicar el formulario 14, a pesar de que no sea el siguiente en el orden secuencial.

FORMULARIO 14 – DEFINICIÓN DE LOS ATRIBUTOS DINÁMICOS DE EJECUCIÓN CONSTANTE

Los formularios 14 y 15 pertenecen al módulo de definición de atributos, pero se sitúa al final de la definición de la instancia, después del módulo de estados y función objetivo, porque requieren de la definición previa de los eventos. Como veremos, para caracterizar los atributos de evento dinámicos se debe haber definido previamente los eventos de la instancia, y por ello se sitúan después del módulo de estados.

El formulario 14, mostrado en la figura 5-19, tiene el objetivo de distinguir entre los atributos dinámicos de ejecución constante y los atributos de evento dinámicos. Recordar que los atributos de ejecución constante son los que reciben el mismo valor para todas las ejecuciones derivadas de un mismo conjunto de ejecuciones; mientras que los atributos de evento dinámico son aquellos que pueden cambiar de valor a lo largo de una ejecución/simulación.

En el formulario 14 se presenta una lista de todos los atributos que fueron definidos como dinámicos en el formulario 8, anteriormente explicado. Los atributos que se quieran definir como atributos de evento dinámico deben ser marcados con la casilla situada a su izquierda. Estos atributos marcados, como pueden cambiar a lo largo de una ejecución, serán caracterizados en el siguiente formulario. Los atributos dinámicos de ejecución constante recibirán un valor en el siguiente nivel, nivel de conjunto de ejecuciones (Process Execution Set). Cabe destacar que los atributos "Clock.Time", "Channel Availability", y "Infrastructure Availability" siempre serán atributos de evento dinámico por exigencias del modelo conceptual, aunque el usuario quiera definirlos como de ejecución constante. Este punto fue explicado en los modelos conceptuales del módulo de atributos. Para pasar al siguiente formulario se seleccionará el botón "Next".

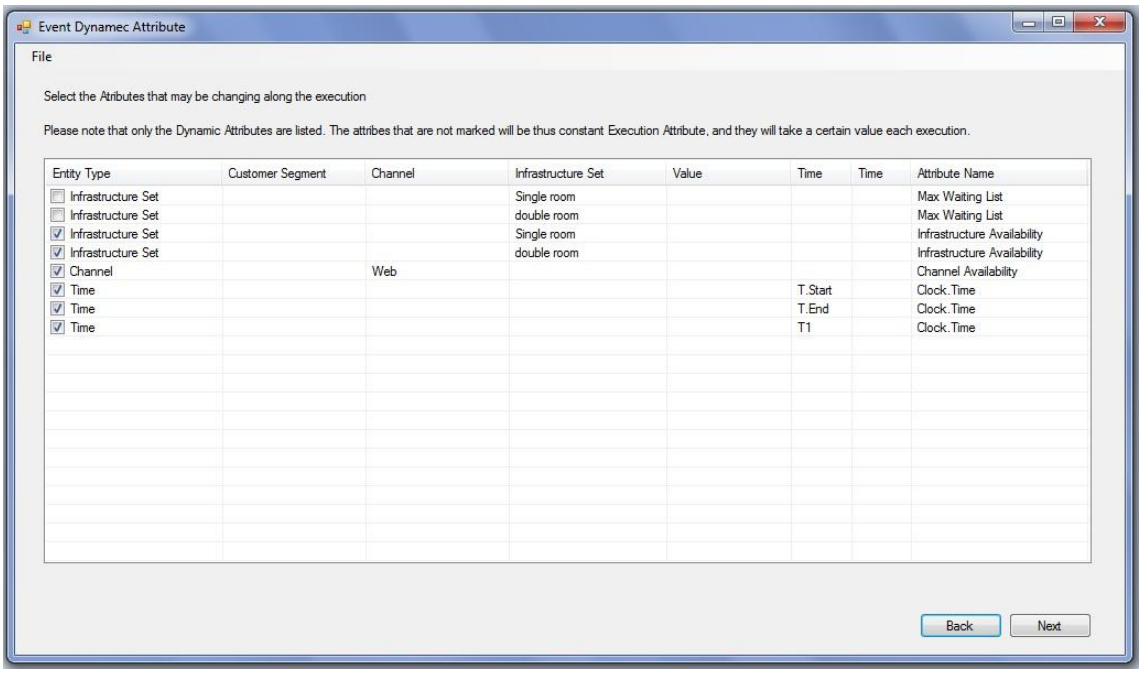


Figura 5-19. Formulario 14, selección de los atributos dinámicos de ejecución constante

FORMULARIO 15 – CARACTERIZACIÓN DE LOS ATRIBUTOS DE EVENTO DINÁMICOS

Una vez definidos qué atributos son de evento dinámico, en el formulario 15 se caracteriza cada uno de estos atributos. Recordar que para caracterizar un atributo de evento dinámico, el cual puede cambiar de valor durante cada ejecución, se deben definir tres aspectos para cada cambio que se pueda producir:

- El evento que ocasiona el cambio. Tanto si es producido por un cambio de estado como por un evento de tiempo, de debe especificar cuál.
- El elemento del modelo perteneciente a la entidad Allocation (Asignación) que desencadena el evento provocando el cambio. Si es un evento de tiempo, no se asocia ningún elemento asignación.
- Qué cambio se produce cuando el elemento del modelo seleccionado desencadena el evento seleccionado. El cambio puede ser un incremento, un decremento, un cambio a un valor determinado, o un cambio en el caso de los atributos booleanos.

Con esto, en la lista superior del formulario 15 mostrado en la figura 5-20, se presentan los atributos de evento dinámico definidos, junto al elemento del modelo asociado a cada atributo. Las tres columnas de la derecha corresponden respectivamente al evento que produce el cambio, el tipo de cambio que se produciría, y la cantidad en caso de ser un cambio de incremento, decremento o cambio a un valor determinado. Estas tres últimas columnas son definidas por el usuario y caracterizan el atributo de evento.

Entre la lista superior y la inferior existe una clara relación, y ésta se expresa por el identificativo presente en la primera columna de ambas listas. De esta manera, las filas de las distintas listas pero con el mismo "ID" están directamente relacionadas, y no están en la misma tabla por motivos de espacio. En la lista inferior del formulario se muestra el elemento de la entidad Allocation que provoca el evento y desencadena el

cambio caracterizado en la lista superior. Por ser un elemento Allocation, está determinado por la combinación de infraestructura, canal, segmento de cliente, espacio temporal, y valor asociado, todo ello mostrado en la lista inferior. Destacar que los atributos caracterizados por eventos de tiempo no tendrán un elemento asignación asociado, y por lo tanto no tendrán ninguna fila asociada en la lista inferior. Esto sucede por ejemplo con el atributo "Clock.Time" (Reloj de la ejecución), que además es un atributo gestionado internamente por la aplicación y no puede ser caracterizado por el usuario aunque se muestre en la lista. Destacar que la lista inferior es sólo útil como display de información para el usuario, pero el éste no puede interaccionar con ella.

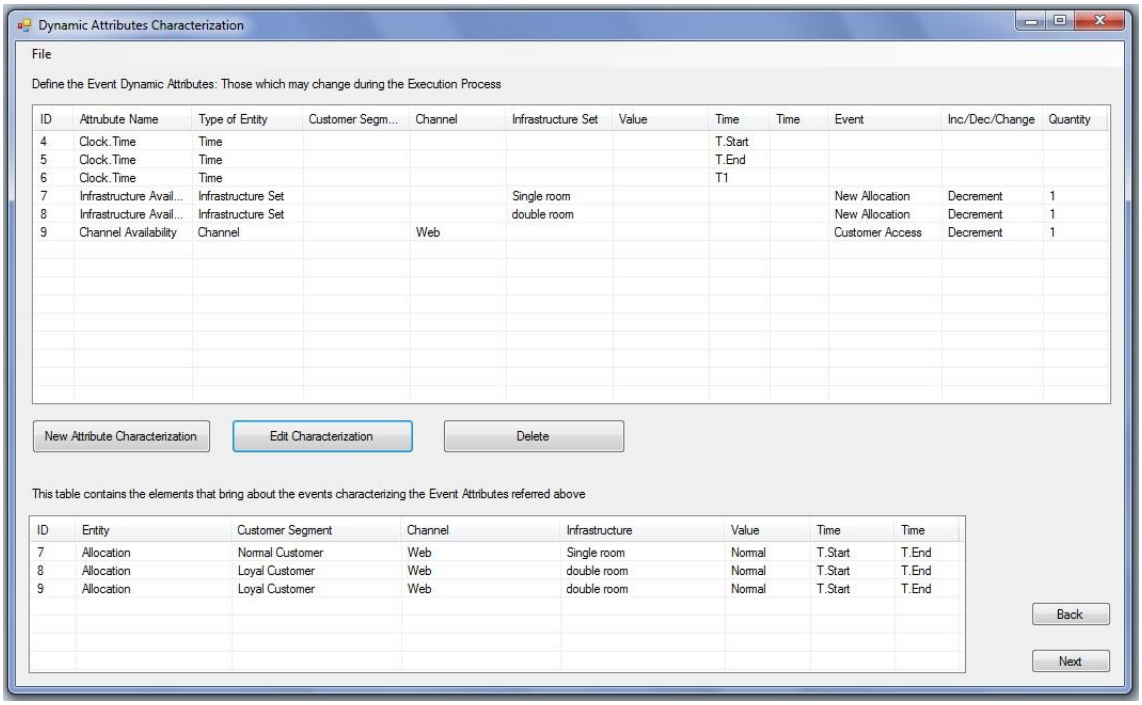


Figura 5-20. Formulario 15, caracterización de los atributos de evento dinámicos

Al cargar por primera vez el formulario, la lista inferior estará vacía, ya que se rellana con la caracterización de cada atributo hecha por el usuario. Vista la relación entre ambas listas, pasamos a explicar los tres botones que se encuentran en mitas del formulario. Estos tres botones están ligados a la lista superior, por lo que las filas que se quieran manipular se deben seleccionar de esta lista:

- *Delete:* Sirve para borrar la caracterización del atributo de evento seleccionado previamente de la lista superior. Sólo se podrá borrar una fila de la lista superior si el atributo implicado ya está caracterizado en otra fila, es decir, como cada atributo de evento debe tener al menos una caracterización (fila), no se podrá borrar si dicho atributo sólo contiene esa fila, y en este caso sólo se podrá editar.
- *Edit Characterization:* Este botón sirve para editar una caracterización, también seleccionada previamente de la lista superior. Al seleccionar una fila de la lista y posteriormente el botón, aparecerá el cuadro de diálogo de la figura 5-21, el cual se explicará a continuación.
- *New Attribute Characterization:* Se selecciona un atributo de la lista superior y al pulsar el botón se genera una nueva línea en dicha lista con un nuevo identificativo (ID), y se muestra el cuadro de diálogo de la figura 5-21 para caracterizar el nuevo evento.

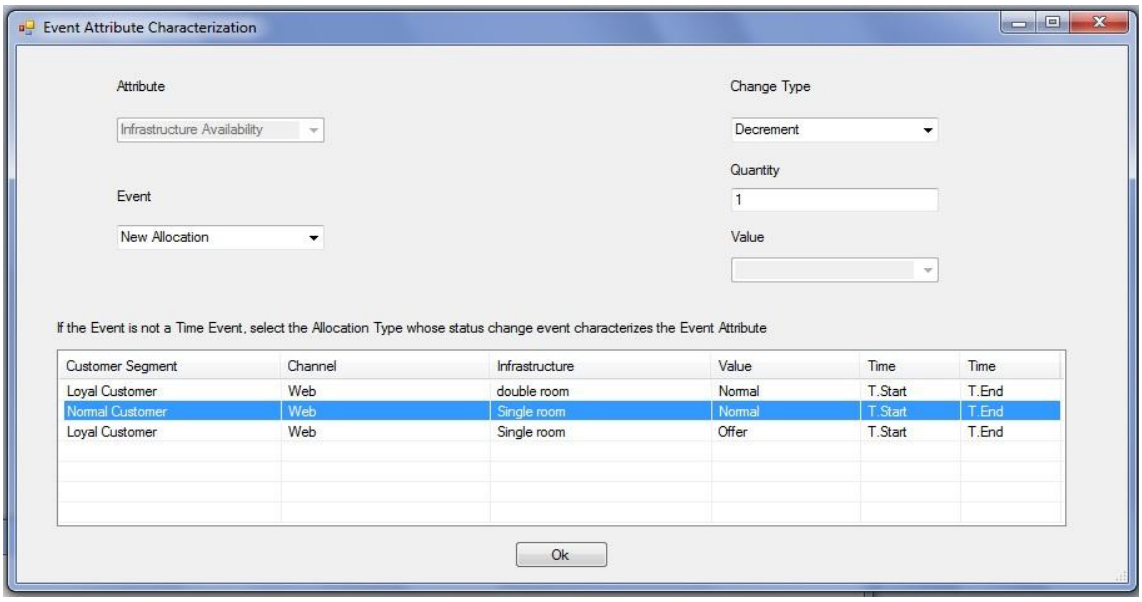


Figura 5-21. Cuadro de diálogo de caracterización de los atributos de evento dinámicos

El cuadro de diálogo de la figura 5-21 sirve para caracterizar el atributo de evento que ha sido seleccionado previamente, tanto para editar como para una nueva caracterización. En la primera casilla de la esquina superior izquierda aparece el atributo seleccionado, el cual no se puede modificar en este cuadro. Debajo se debe seleccionar el evento, de una lista desplegable, que desencadena el cambio. En la lista inferior aparecerán todos los elementos asignación definidos en el modelo, y se deberá seleccionar uno de ellos. En la casilla de la esquina superior derecha se seleccionará el tipo de cambio que se producirá, incremento, decremento, cambio, o cambio a. Dependiendo de la opción elegida en esta última casilla y del tipo de atributo seleccionado se activará una de las dos casillas por debajo. Si el atributo no es booleano, se activará la casilla "Quantity" si el cambio es un "incremento", "decremento" o "cambio a", y se especificará la cantidad, ya que la opción sólo "cambio" no está permitida para este tipo de atributos. Si el atributo es booleano, sólo se permiten las opciones "cambio a" y "cambio", y se activará la casilla "Value" en el primer caso. Una vez definidos los tres aspectos necesarios (Evento, elemento asignación, y cambio), se pulsa ok y la caracterización se reflejará en el formulario 15 de la figura 5-20.

Por último, se disponen de los botones "Back" para volver al formulario anterior, y "Next" si se ha completado la caracterización y se desea continuar al siguiente paso. Antes de continuar, la aplicación comprueba que todos los atributos de eventos han recibido al menos una caracterización de evento, en cuyo caso pasaremos a la ventana de finalización de la instancia de proceso y que explicaremos más adelante.

5.2.2 MÓDULO DE DEFINICIÓN DE ESTADOS

En este apartado pasamos a describir los formularios de la aplicación destinados a la definición de los estados y la caracterización de eventos de cada instancia de proceso. Para ellos, se explicarán los formularios 10, 11, y 12, y algún cuadro de diálogo.

Tras el formulario 9, de definición de los valores de los parámetros de atributos estáticos explicado anteriormente, pasamos al formulario 10 y comienza el módulo de definición de estados de la instancia.

FORMULARIO 10 – DEFINICIÓN DE ESTADOS Y CAMBIOS DE ESTADO

El objetivo del formulario 10 mostrado en la figura 5-22 es la definición de los estados y los diferentes cambios de estados que conforman el árbol de estados de la instancia de proceso, pudiendo tener lugar en una ejecución.

En la lista de la parte izquierda del formulario se muestran los estados definidos. En esta lista aparecerán tanto los estados fijados por el modelo conceptual, es decir, los estados necesarios en toda instancia, como los estados adicionales definidos por el usuario. El la figura 5-22 aparecen los estados necesarios que serán cargados por la aplicación en cualquier instancia. Con los botones de la parte derecha de la lista se puede modificar dicha lista, a excepción de los estados fijados que no pueden ser modificados. Mediante el botón "Remove" se pueden borrar los estados seleccionados previamente. El botón "Edit" sirve para editar el nombre de un estado seleccionado, y se hará mediante un cuadro de diálogo. Por su parte, el botón "Add" sirve para añadir un nuevo estado a la lista, asignándole un nombre que no coincida con el de ningún otro estado.

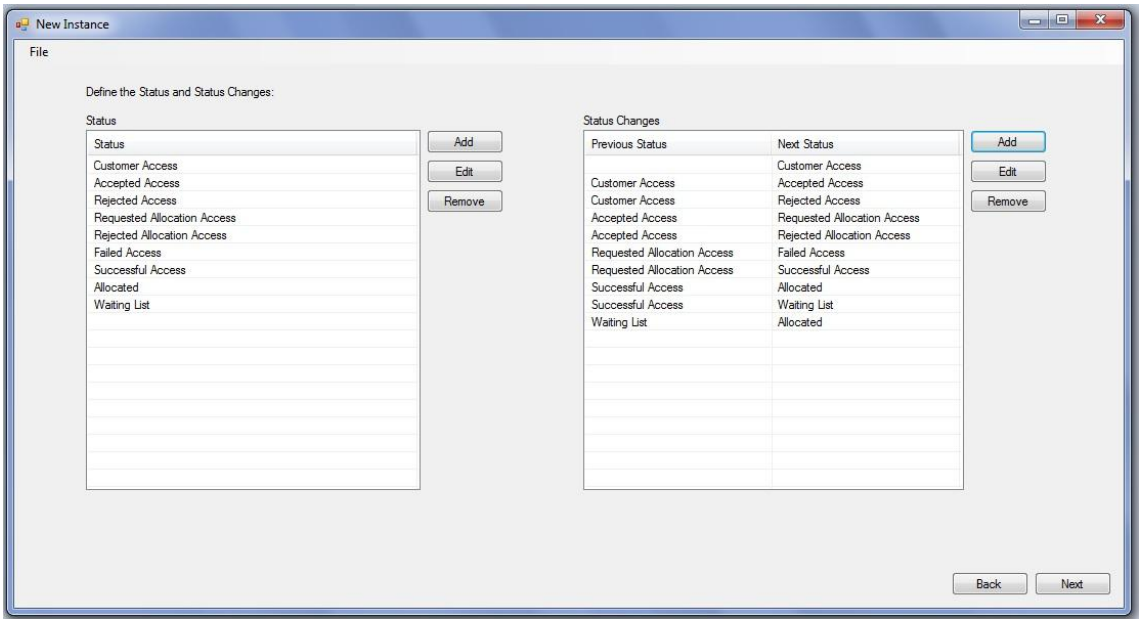


Figura 5-22. Formulario10, definición de los estados y cambios de estado de la instancia

De la misma manera, la lista de la derecha muestra los cambios de estado fijados por el modelo conceptual, y los cambios de estado que sean definidos por el usuario. Al igual que con los estados, los cambios de estado fijados no se pueden modificar, y son los mostrados en la figura 5-22. La columna de la izquierda muestra el estado previo, y la de la derecha el estado siguiente al que se avanza. Los botones de la derecha sirven para borrar un cambio de estado mediante el botón "Remove", editar un cambio de estado mediante el botón "Edit", o con el botón "Add añadir un nuevo cambio de estado. Tanto para editar como para añadir, la aplicación nos mostrará el cuadro de diálogo de la figura 5-23.

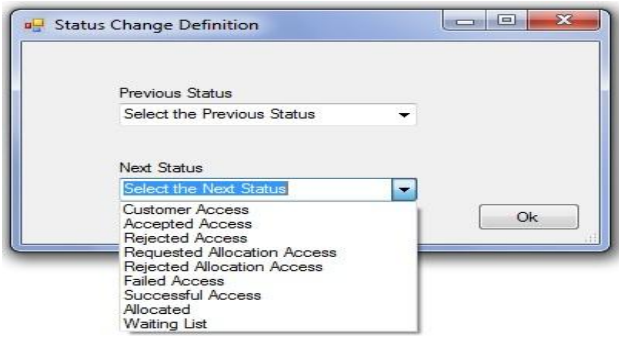


Figura 5-23. Cuadro de diálogo de definición de un cambio de estado

En este cuadro de diálogo hay que especificar el estado previo (Previous Status) y el estado siguiente (Next Status), a través de sendas casillas desplegables que sólo posibilitan la elección de estados ya definidos, es decir, de estados listados en el formulario 10. La aplicación no permitirá que se añada un cambio de estado ya existente.

Si se borra un estado, se borrarán los cambios de estado que contengan dicho estado, así como los eventos asociados. Como se explicó en los modelos conceptuales, todo cambio de estado supone un evento, por lo que al borrar un cambio de estado se borrará el evento asociado. En el siguiente formulario se tratan los eventos, en consecuencia hay una relación directa entre ambos formularios.

Como siempre, se dispone de los botones "Back" y "Next", y con este último pasaremos al formulario 11.

FORMULARIO 11 – CLASIFICACIÓN DE EVENTOS DE LA INSTANCIA

Como se explicó en los modelos conceptuales, los eventos están compuestos por los cambios de estado definidos, y por los eventos de tiempo (Marcados por los elementos del modelo de la entidad Time). Por lo tanto, los eventos ya han sido definidos en pasos anteriores del proceso de definición, y en el formulario 11 de la figura 5-24 el objetivo es definir la clasificación de todos los eventos que caracterizan la instancia.

En el formulario 11 se presenta una lista con todos los eventos definidos en la instancia. En la figura 5-24 se pueden apreciar los eventos cuya clasificación no puede ser modificada, pues ni los eventos de tiempo ni los eventos de cambio de estado fijados por la herramienta pueden sufrir alteraciones de clasificación. En resumen. Sólo se podrá editar la clasificación de aquellos eventos de cambio de estado que hayan sido credos por el usuario. En la lista se presenta el nombre de cada evento, las características que lo definen, y la clasificación de los mismos. Existen dos columnas de clasificación porque cada evento de cambio de estado puede estar caracterizado hasta por dos atributos, y por ello puede tener una clasificación distinta por atributo.

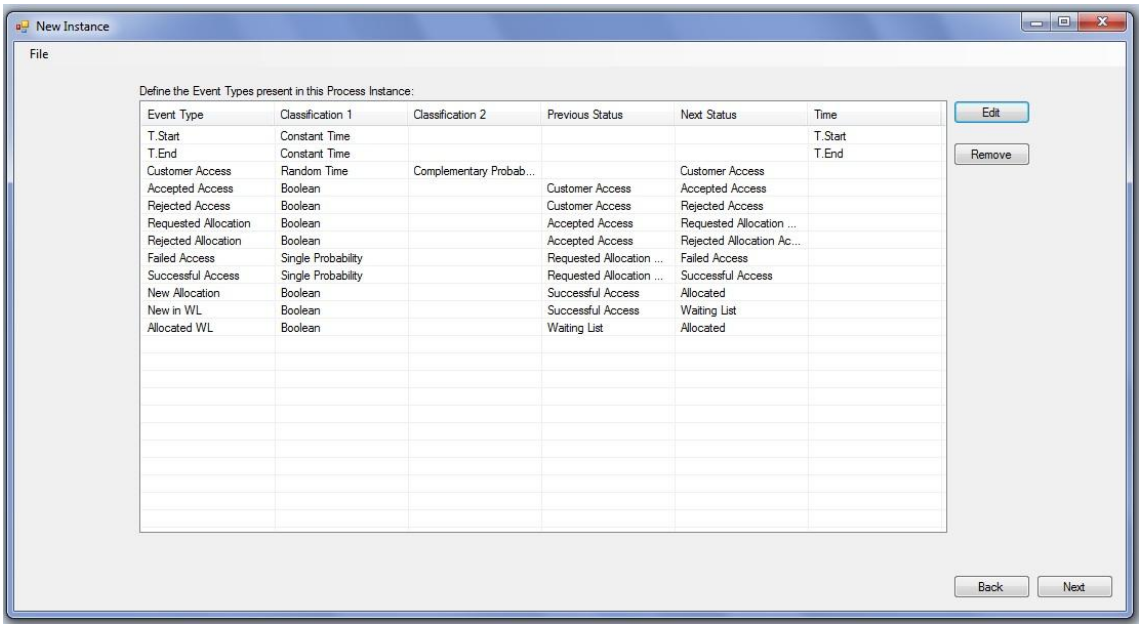


Figura 5-24. Formulario11, clasificación de eventos de la instancia de proceso

Existen dos botones a la derecha de la lista, uno para borrar el evento seleccionado (Botón "Remove"), y otro para editar la clasificación del evento preseleccionado ("Edit"). Al seleccionar un evento y el botón "Edit", se mostrará el cuadro de diálogo de la figura 5-25; en el que se recoge el nombre del evento, las características del cambio de estado, y el usuario debe definir la/s clasificaciones mediante las casillas desplegables. Esta definición se incorporará a la lista del formulario.

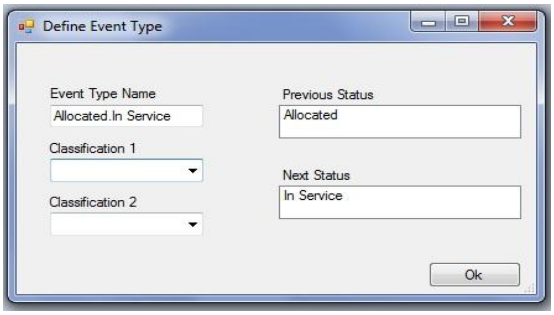


Figura 5-25. Cuadro de diálogo de clasificación de un evento

Con el botón "Next" se pasará al formulario 12, siempre que estén definidas las clasificaciones de todos los eventos.

FORMULARIO 12 – CARACTERIZACIÓN DE EVENTOS

El formulario 12 es el último en la definición del módulo de estados de la instancia. En este formulario, mostrado en la figura 5-26, el objetivo es definir las condiciones que llevan a producirse el evento, es decir, el atributo o los atributos que caracterizan el evento, así como los valores o condiciones que lo desencadenan.

En la lista de la figura 5-26 podemos ver que aparecen todos los eventos definidos, identificados por el nombre, y junto a ellos se muestra el atributo 1 que gobierna el evento y el valor que debe adquirir. Se incluye la posibilidad de añadir un atributo 2 y su valor, ya que como se ha comentado un evento puede estar caracterizado hasta por

dos atributos de manera que se deben cumplir ambas condiciones. Una vez más, los eventos de tiempo al igual que los eventos de cambio de estado fijados por el modelo no pueden ser editados. En la figura 5-26 se muestra la caracterización de todos los eventos de cambio de estado fijados, y que la aplicación cargará en cualquier instancia.

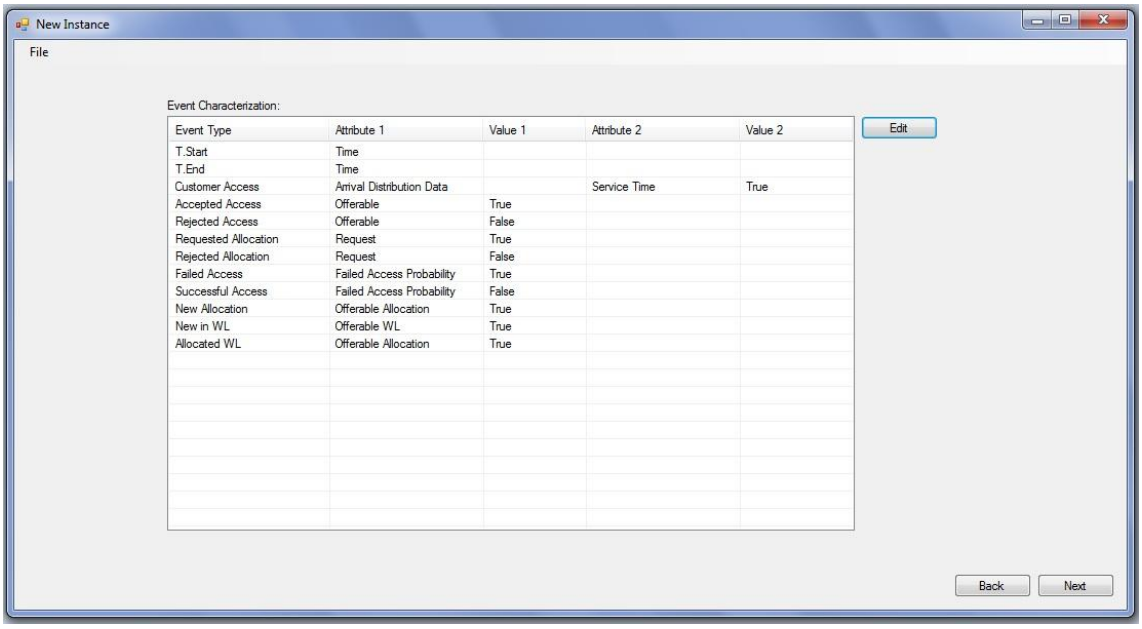


Figura 5-26. Formulario12, caracterización de eventos de la instancia de proceso

Para editar los eventos de cambio de estado definidos por el usuario, podemos usar el botón "Edit" que se sitúa junto a la lista. Seleccionando el evento deseado y posteriormente el botón "Edit", se mostrará el cuadro de diálogo de la figura 5-27.

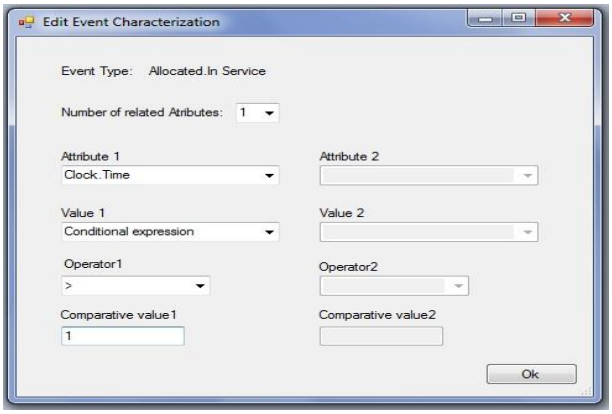


Figura 5-27. Cuadro de diálogo de caracterización de un evento

El nombre del evento aparecerá en la parte superior. Luego se debe especificar cuántos atributos caracterizan el evento, y en caso de ser dos se habilitarán las casillas de la columna de la derecha, sino sólo podremos definir las de la columna izquierda como en el ejemplo de la figura 5-27. Seguidamente se define qué atributo gobierna el evento, así como el valor que debe adoptar.

Una funcionalidad muy potente que incluye el proyecto es la posibilidad de que un evento esté gobernado por atributos dinámicos. Al ser atributos que pueden cambiar de valor durante la simulación, el evento puede estar gobernado por una condición regida

por un operador relacional. En el ejemplo de la figura vemos que el evento "Allocated.In Service" está caracterizado por el reloj de la ejecución (Clock.Time), y se dará el evento cuando dicho atributo adquiera un valor superior a 1, como expresa la condición y el valor de comparación. Esta funcionalidad por la que un evento está caracterizado por una condición de relación, sólo puede darse si el atributo seleccionado es dinámico.

Una vez que todos los eventos han sido caracterizados, se puede acceder al siguiente formulario mediante el botón "Next", pasando a la definición de las funciones objetivo.

5.2.3 MÓDULO DE DEFINICIÓN DE FUNCIÓN OBJETIVO

Con este apartado se concluye la definición de los módulos de la instancia. En este apartado se describe la parte de la aplicación destinada a la definición de las funciones objetivo que definen cada instancia del proceso. Para ello, se explica el formulario 13 y sus aspectos más importantes.

FORMULARIO 13 – DEFINICIÓN DE FUNCIONES OBJETIVO

A través del formulario 13, mostrado en la figura 5-28, se definen todas las funciones objetivo que caractericen la instancia. Tanto las funciones de maximización, como las de minimización, o las expresiones lógicas.

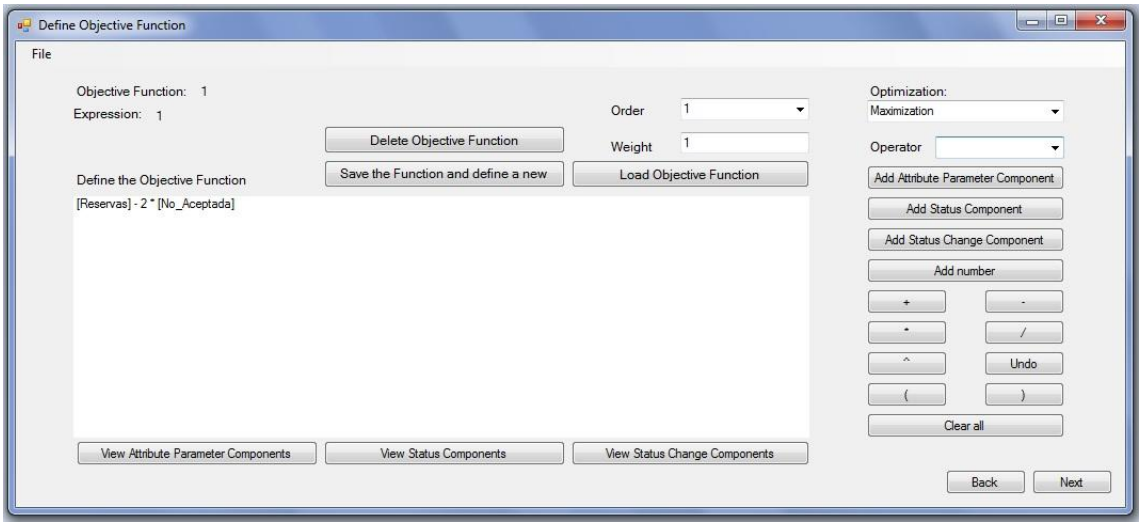


Figura 5-28. Formulario13, definición de funciones objetivo

En primer lugar, en la esquina superior izquierda se muestra el número de la función objetivo en definición, si se carga una ya definida aparecerá aquí el nombre, y debajo el número identificativo de expresión, ya que recordemos que las funciones objetivo están compuestas por expresiones matemáticas. En la parte superior derecha se definen tanto el orden de prioridad de la función (Order), como el peso (Weight), el tipo de optimización deseado (Maximización, minimización, o lógoca), y si la función es lógica se debe definir el operador (Operator) que relacionará las dos expresiones que componen la función lógica.

Puesto que nuestro modelo conceptual contempla la posibilidad de definir muchas funciones objetivo ordenadas por un nivel de prioridad, en la parte superior de la pantalla central hay tres botones para gestionar las distintas funciones objetivo. El primer botón es "Delete Objective Function", y sirve para borrar una de las funciones objetivo de una lista que mostrará todas las funciones guardadas. Existe otro botón para salvar la función objetivo que está siendo definida y abrir una nueva para ser definida (Save the Function and define a new), tras pulsar el botón habrá que asignar un nombre identificativo a cada función. Si la función es lógica, mediante este botón se pasará a la definición de la segunda expresión para completar la función. El tercer botón sirve para cargar una función objetivo de la lista de funciones guardadas (Load Objective Function).

La pantalla central sirve de display en la que se mostrarán los términos de cada función y los operadores que los vinculan. A la derecha de dicha pantalla se presentan una serie de botones orientados a la definición y edición de las funciones objetivo:

- Tres botones que dan la posibilidad de añadir componentes dinámicos a la función objetivo. Cada uno de ellos da paso a un cuadro de diálogo a través del cual se define el componente y sus propiedades. Cada botón está asociado a un tipo de componente dinámico: componente de parámetro de atributo (Add Attribute Parameter Component) asociado a un atributo de cualquier tipo, componente de estado (Add Status Component) asociado a un estado, y componente de cambio de estado (Add Status Change Component) asociado a un cambio de estado.
- Un botón para incluir números en la función a través de un cuadro de diálogo (Add Number)
- Cinco botones de signos de operación: * (Multiplicación), / (División), ^ (Potencia), + (Suma), y - (Resta). Más dos botones de abrir y cerrar paréntesis.
- Un botón para deshacer la última acción realizada (Undo).
- Un botón para borrar toda la función objetivo en curso (Clear all).

Respecto a los botones de adición de componentes dinámicos, suponen un punto clave en la definición de toda función objetivo. Al seleccionar cada uno de los tres botones se abrirán cuadros de diálogo distintos, pero muy similares. Para explicar las reglas en las que se basa la definición de cada componente, se va a explicar a través de la figura 5-29 que muestra un ejemplo de definición de un componente dinámico de cambio de estado, a través del cuadro de diálogo correspondiente.

En primer lugar se asigna un nombre al componente, en este caso "No Aceptada". Seguidamente se selecciona, a través de casillas desplegables, el estado previo y el estado siguiente al que se referirá el cambio de estado. Para un cambio de estado se necesita definir qué elemento asignación lo desencadena. Primero se habilitará la casilla correspondiente a Customer Segment. Se pude escoger entre un segmento de cliente concreto, o seleccionar la opción "All", que permitirá hacer un sumatorio de los elementos asignación con cualquier segmento de cliente. Si selecciono un segmento concreto, se habilitará la casilla canal con los canales que dicho segmento pueda emplear, y siempre con la opción "All". Así, sucesivamente se irán habilitando las casillas hasta definir el/los elementos asignación que desencadenan el cambio de estado y se quieren considerar en la función objetivo. Si por ejemplo se selecciona "All" en Infrastructure Set, el componente refleja la suma de las aportaciones de todos los

elementos asignación que contengan cualquier infraestructura, y que sufren el cambio de estado definido

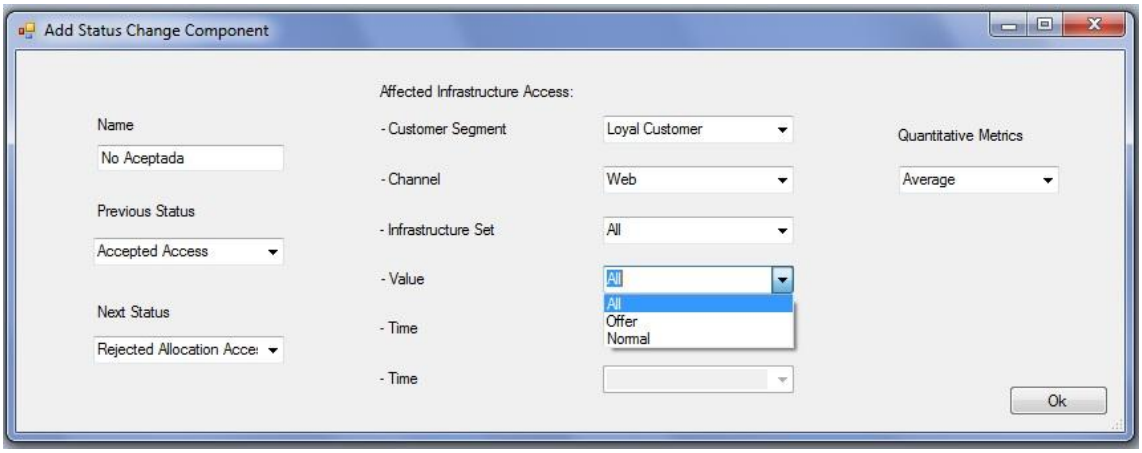


Figura 5-29. Cuadro de diálogo de adición de un componente dinámico de cambio de estado

En el caso de los componentes de atributo, para definir el/los elementos implicados, se habilitarán las casillas dependiendo de a qué entidad este vinculado el atributo seleccionado. Es decir, si selecciono un atributo asociado a la entidad "Channel", sólo se habilitará la casilla "Channel" de la columna de la derecha. En el caso de un componente de estado, también habrá que definir el/los elementos asignación, ya que por los distintos estados pasarán elementos asignación concretos durante la ejecución del proceso.

La columna de la derecha contendrá una o dos casillas dependiendo del tipo de componente dinámico, y los valores que se pueden seleccionar fueron explicados en los modelos conceptuales de la función objetivo (Apartado 3.4.3).

Por último, queda explicar los tres botones de la parte inferior de la ventana central de la figura 5-28. Se trata de botones, cada uno asociado a un tipo de componente dinámico, mediante los cuales la aplicación muestra sendas ventanas de consulta que presentan una lista de todos los componentes de cada tipo definidos por el usuario en la función objetivo en curso. En la figura 5-30 se observa la ventana de consulta correspondiente a componentes de tipo cambio de estado, a la cual se accederá pulsando el botón “View Status Change Components”. Como se ve en la figura, se muestra tanto el nombre del componente, como el cambio de estado definido, y el/los elementos asignación involucrados.

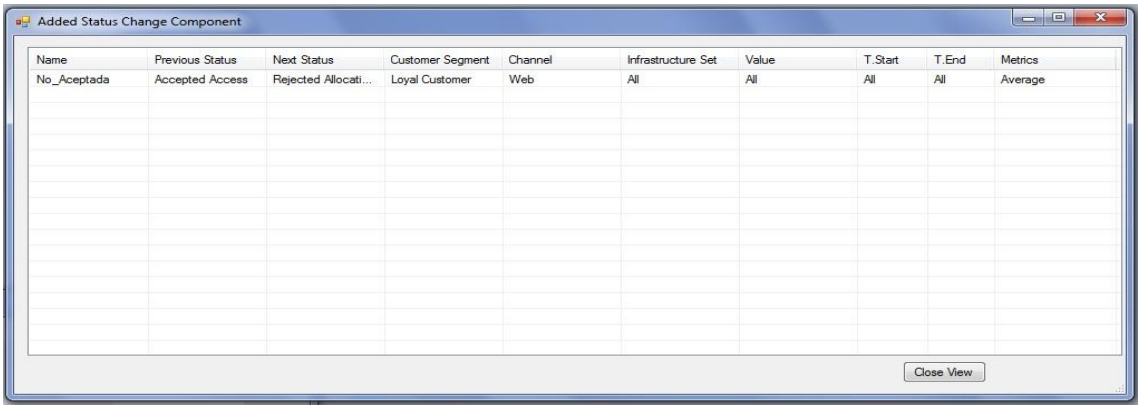


Figura 5-30. Cuadro de consulta de componentes dinámicos de cambio de estado

FINALIZACIÓN DE LA DEFINICIÓN DE INSTANCIA

Una vez definido todos los módulos de la instancia, pasaremos a cerrar el proceso de definición de la misma. Recordar que aunque el último formulario explicado es el 13, perteneciente a la función objetivo, la definición de la instancia finaliza en el formulario15, perteneciente al módulo de atributos y por motivos ya explicados.

Por tanto, tras definir el formulario 15 y pasar al siguiente paso, la aplicación nos mostrará la ventana de la figura 5-31. En ella se ofrecen cinco opciones: borrar la instancia (Discard the Instance), salvar la instancia (Just save the Instance), salvar la instancia como plantilla (Save the Instance as Template), salvar la instancia como plantilla por defecto (Save the Instance as Default Template), o salvar la instancia y definir un conjunto de ejecuciones (Save the Instance and define an Execution Set). Al salvar la instancia como plantilla también se salvará la instancia en sí misma.

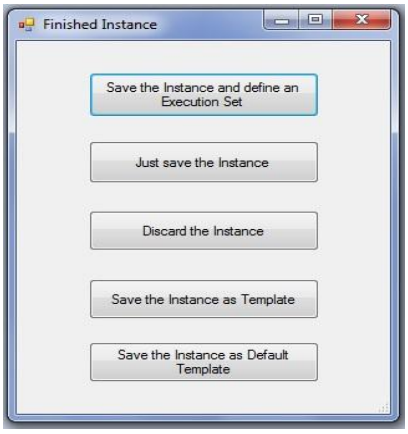


Figura 5-31. Cuadro de finalización de la definición de la instancia

CAPÍTULO 6

EJEMPLO DE USO DE LA HERRAMIENTA

Coma ya se han explicado los tres pilares fundamentales del proyecto, como son el modelo conceptual (Capítulo 3), el modelo de datos (Capítulo 4), y la aplicación (Capítulo 5), este capítulo estará dedicado a presentar un ejemplo de aplicación de la herramienta informática. El ejemplo se basa en una unidad de cardiología de urgencias, perteneciente al sector sanitario, e incluirá tanto la definición del problema de asignación de infraestructuras, a través de la definición del modelo y de una instancia del proceso, como la definición del conjunto de ejecuciones y una simulación completa del proceso.

El capítulo estará estructurado en dos bloques. El primer apartado constará de la definición del modelo y la instancia del proceso, partes que cubre el alcance de la aplicación implementada, y la segunda parte que recogerá la definición del nivel conjunto de ejecuciones y la simulación/ejecución de un proceso, realizada con Microsoft Excel.

6.1 UNIDAD DE CORDIOLOGÍA DE URGENCIAS: DEFINICIÓN DEL MODELO Y UNA INSTANCIA DEL PROCESO

Un hospital se plantea reestructurar la Unidad de cardiología de Urgencias de manera que se pueda aumentar la eficiencia, reduciendo costes innecesarios, para afrontar la disminución del presupuesto para años próximos. Por tratarse del sector sanitario, no se desea que el aumento de eficiencia buscado perjudique a la calidad del servicio. En este punto, los gestores del hospital deciden emplear nuestra herramienta para plantear y resolver el problema.

Un analista del grupo de gestión decide ejecutar RM Modeler 2.0, y comienza la definición del modelo de proceso. En primer lugar comienza creando un nuevo modelo y asignándole un nombre y una fecha (Figura 6-1).

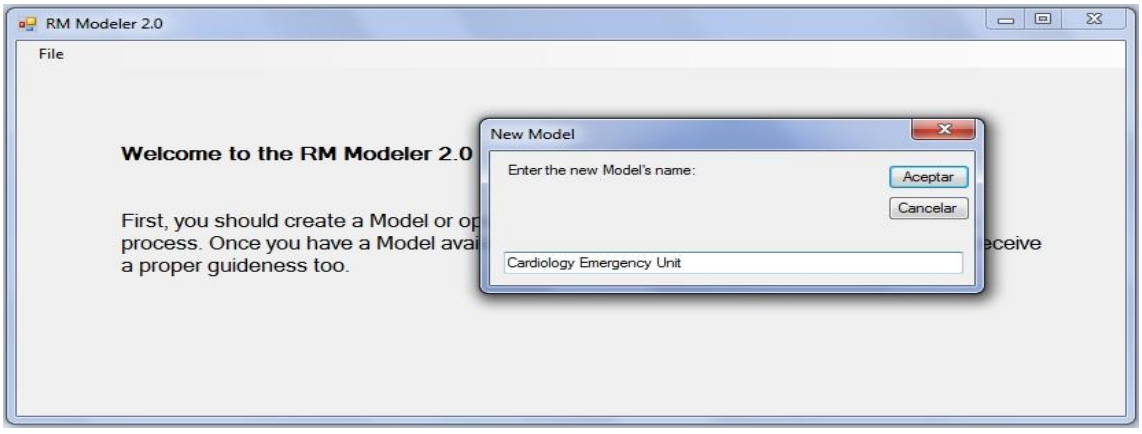


Figura 6-01. Ejemplo Sector Sanitario: Creación nuevo modelo

En el siguiente paso, el asistente define los distintos elementos de las entidades segmento de cliente, canal, infraestructura, y valor, mostrado en la figura 6-2. Los clientes se refieren a los pacientes, y han sido segmentados en función de la gravedad. Se definen dos segmentos de clientes, los levemente enfermos (Slightly ill), y los seriamente enfermos (Seriously ill). Respecto al canal de acceso, los pacientes pueden acceder a la Unidad de Cardiología por ambulancia (Ambulance), o por su propio pie por recepción (Reception). Las infraestructuras son en este caso los médicos, y se

dispone de médicos cardiólogos (Cardiologist), y médicos internos residentes (Medical Intern Resident), que se encuentran en proceso de formación. Como valores, se define un valor alto (High) y una bajo (Low), y se emplean como medida del beneficio social que supone llevar a cabo con éxito la atención de un paciente de una forma u otra. Se trata de valores abstractos, difíciles de cuantificar a la hora de asignar un valor numérico.

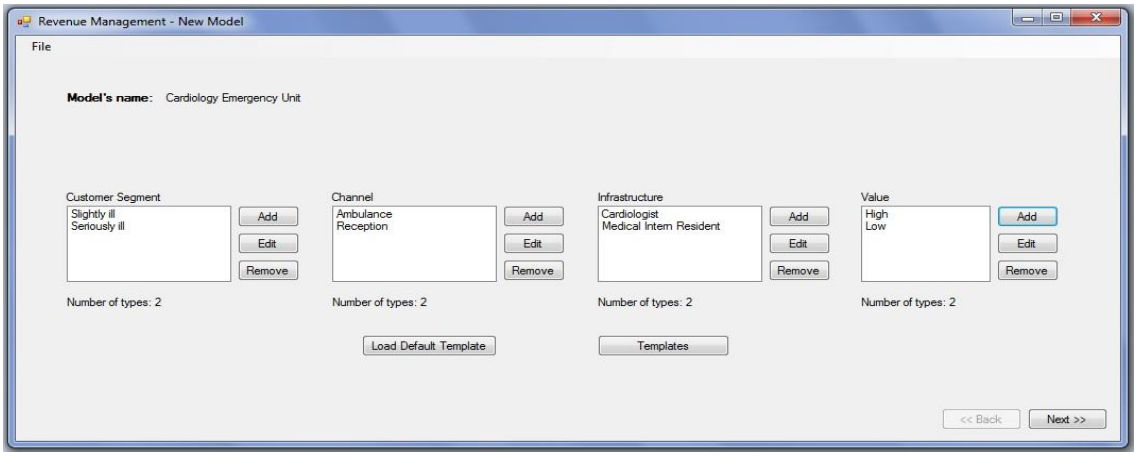


Figura 6-02. Ejemplo Sector Sanitario: Definición de Elementos del modelo

El siguiente paso es definir los elementos de la entidad Infrastructure Access que se permiten en el modelo, es decir, las combinaciones de los elementos segmento de cliente, canal, e infraestructura. Estas combinaciones se muestran en la figura 6-3 y 6-4.

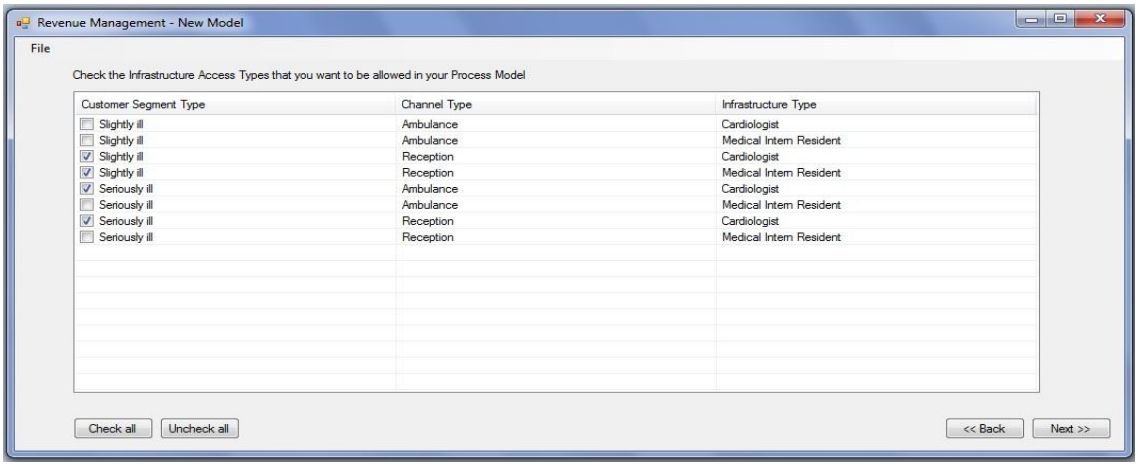


Figura 6-03. Ejemplo Sector Sanitario: Definición de Elementos de Infrastructure Access

Mirando a la elección de elementos de infrastructure Access, se debe destacar alguna observación. Si el paciente es levemente enfermo, éste siempre accederá por recepción (canal), de manera que se reserva la ambulancia para algunos casos de pacientes graves. Por otro lado, se restringe la posibilidad de que un paciente seriamente enfermo sea atendido por un médico residente, pues estos pacientes requieren de la experiencia y cualificación de un cardiólogo. Con esto, tenemos cuatro posibles accesos a infraestructuras.

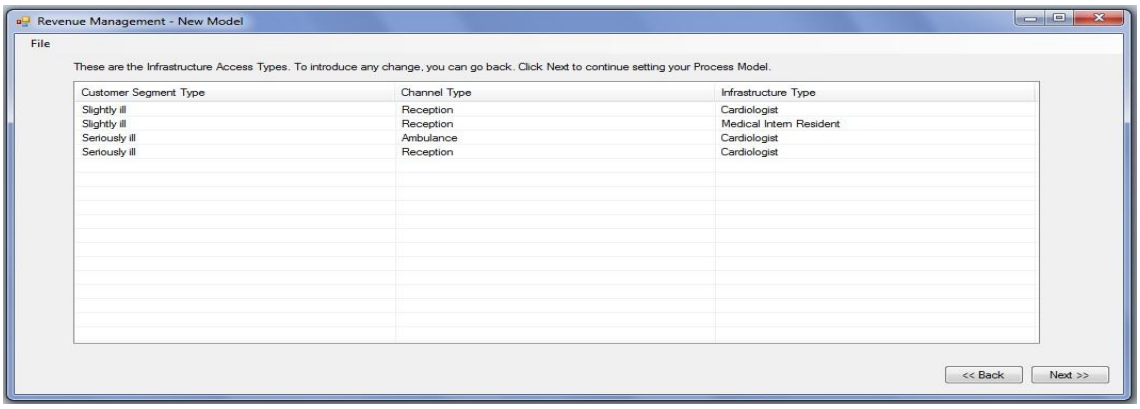


Figura 6-04. Ejemplo Sector Sanitario: Listado de Elementos de Infrastructure Access

Para los elementos de tiempo, los únicos requerimientos son un tiempo inicial (T.Start) y un tiempo final (T.End) de simulación, sin tiempos intermedios (figura 6-5).

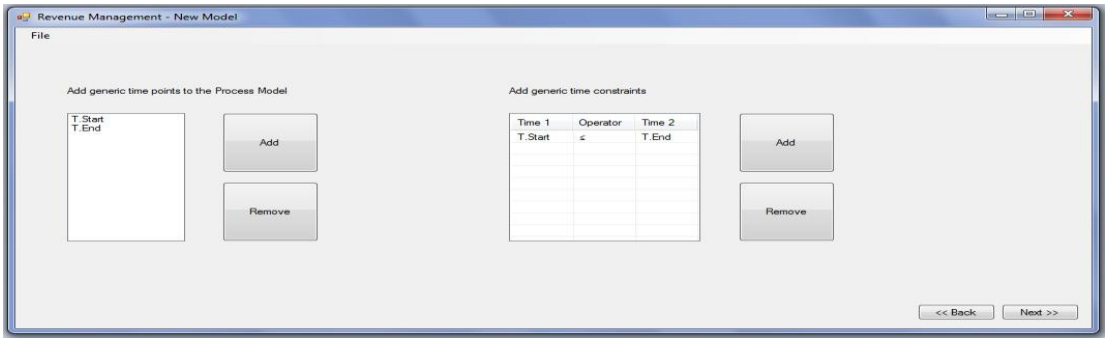


Figura 6-05. Ejemplo Sector Sanitario: Definición de elementos de tiempo y restricciones

En el último paso de definición del modelo, se definen los elementos de asignación (Allocation). En la figura 6-6 se muestran los cuatro elementos allocation definidos, que provienen cada uno de un elemento Infrastructure Access, y con un valor alto asociado a excepción de los clientes leves que son atendidos por un cardiólogo. Esto se debe a que atender satisfactoriamente a un paciente grave siempre genera un valor social alto, mientras que un paciente leve es socialmente más valioso si contribuye a la formación de un médico interno residente, pues en estos pacientes un cardiólogo no es necesario.

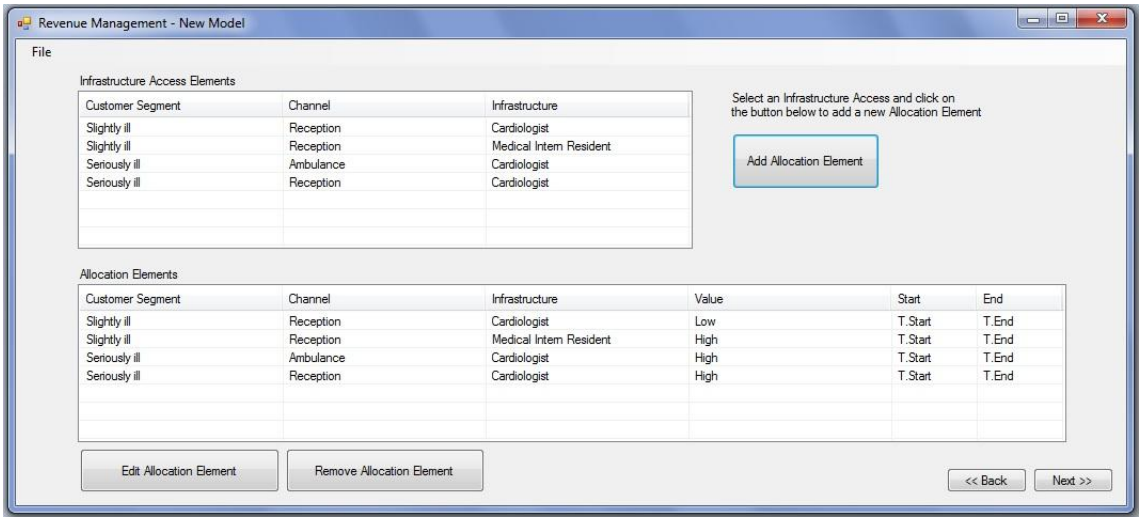


Figura 6-06. Ejemplo Sector Sanitario: Definición de elementos de Allocation

Una vez definido el modelo de proceso, pasamos a definir la instancia. El objetivo de la instancia es un análisis de necesidades de personal, para, como hemos introducido al principio, ajustar costes sin reducir la calidad del servicio. La instancia se llamará "Personnel Analysis" (Análisis de personal).

En primer término pasamos a definir los atributos necesarios y que se muestran en la figura 6-7. Los atributos fijados por el modelo ya han sido explicados, por lo que en la tabla 6-1 figuran los añadidos por el analista del hospital (Usuario) y pertenecientes a la entidad Infrastructure Access.

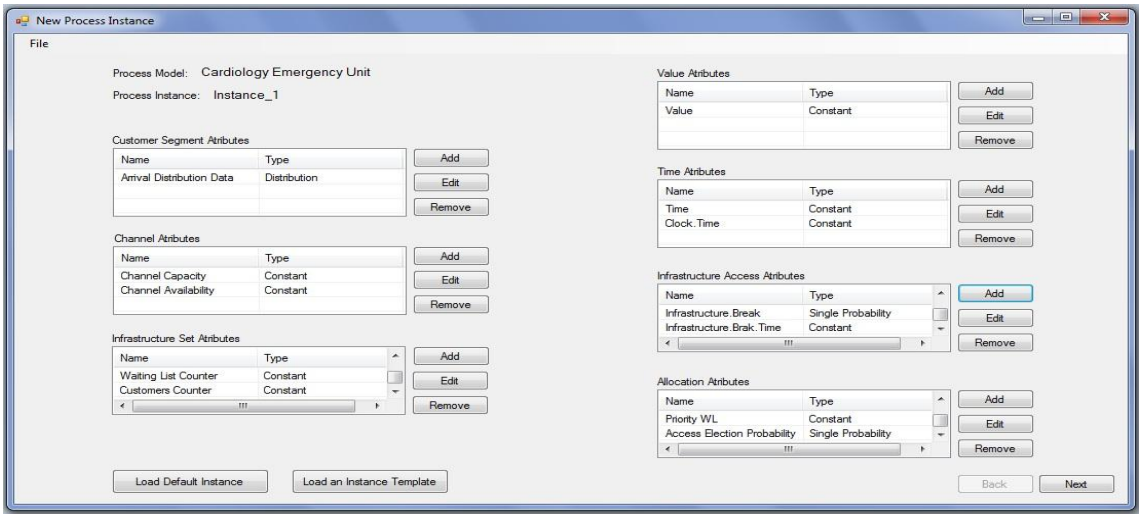


Figura 6-07. Ejemplo Sector Sanitario: Definición de Atributos

	Nombre Atributo	Tipo	Significado
Infrastructure Access	Time.Allocated.InService	Constant	Tiempo entre estado Allocated y In Service
	Time.Served.Closed	Constant	Tiempo entre estado Served y In Closed
	Time.Failed.Closed	Constant	Tiempo entre estado Failed y Closed
	Time.Cancelled.Closed	Constant	Tiempo entre estado Cancelled y Closed
	WL Failed	Probability	Probabilidad de que paciente en lista espera muera
	WL Cancelled	Probability	Probabilidad de que paciente en lista espera se marche
	Break	Probability	Probabilidad de que un paciente en lista de espera se tome un descanso (Ej: cafetería)
	Break.Time	Distribution	Tiempo de descanso del paciente en lista de espera
	Allocated.Cancelled	Probability	Probabilidad de que un paciente en estado Allocated se marche
	Allocated.Failed	Probability	Probabilidad de que un paciente en estado Allocated muera
	Inservice.Failed	Probability	Probabilidad de que un paciente en servicio muera
	InService.Interrupted	Probability	Probabilidad de que un servicio se interrumpa
	Interruption.Failed	Probability	Probabilidad de que un paciente cuyo servicio está interrumpido muera
	Interruption.Time	Constant	Tiempo de interrupción de servicio
	Infrastructure.Break	Probability	Probabilidad de que un médico se tome un descanso
	Infrastructure.Break.Time	Constant	Tiempo de descanso del médico

Tabla 6-1. Ejemplo Sector Sanitario: Nuevos Atributos de Infrastructure Access

Además, existen dos nuevos atributos pertenecientes a la entidad Infrastructure, que serán "WL Counter" (Contador de lista de espera) y "Clients Counter" (Contador de clientes). Son definidos como atributos de evento dinámicos, pues sirven para contar los pacientes de la lista de espera (Va cambiando) y pacientes asignados a un médico respectivamente, y su valor irá cambiando a lo largo de la simulación.

Las variables definidas en el problema son el número de médicos cardiólogos y médicos internos residentes, pues se trata de dimensionar el personal para reducir costes sin reducir eficiencia. El atributo es "Quantity", que representa la cantidad de médicos. Por su parte, entre los atributos dinámicos definidos (Figura 6-8), se encuentran los ya mencionados "WL Counter" y "Clients Counter", a parte de los atributos dinámicos fijados por el modelo conceptual, explicados en el capítulo 3, como "Infrastructure Availability" o "Channel Availability", o algunos atributos de la entidad Allication como "Offerable".

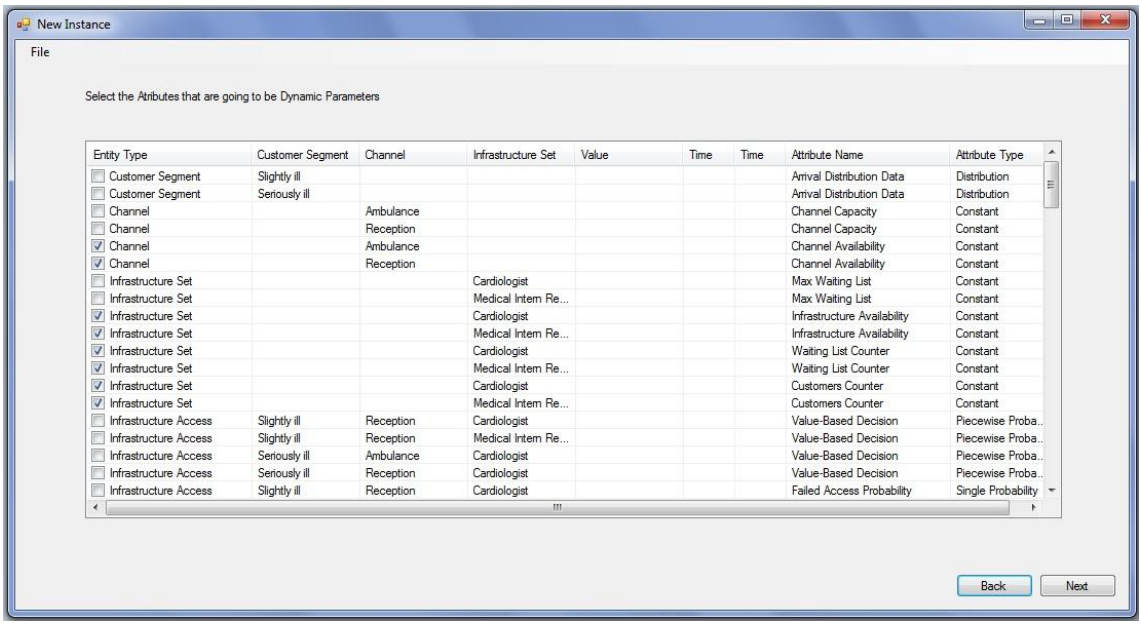


Figura 6-08. Ejemplo Sector Sanitario: Definición de Atributos dinámicos

En el siguiente paso se asignan valores a todos aquellos atributos que no son ni variables ni dinámicos, es decir, a los atributos estáticos de la instancia (Figura 6-9). En nuestro ejemplo, el analista emplea datos históricos de la Unidad, tomando como unidad temporal la hora.

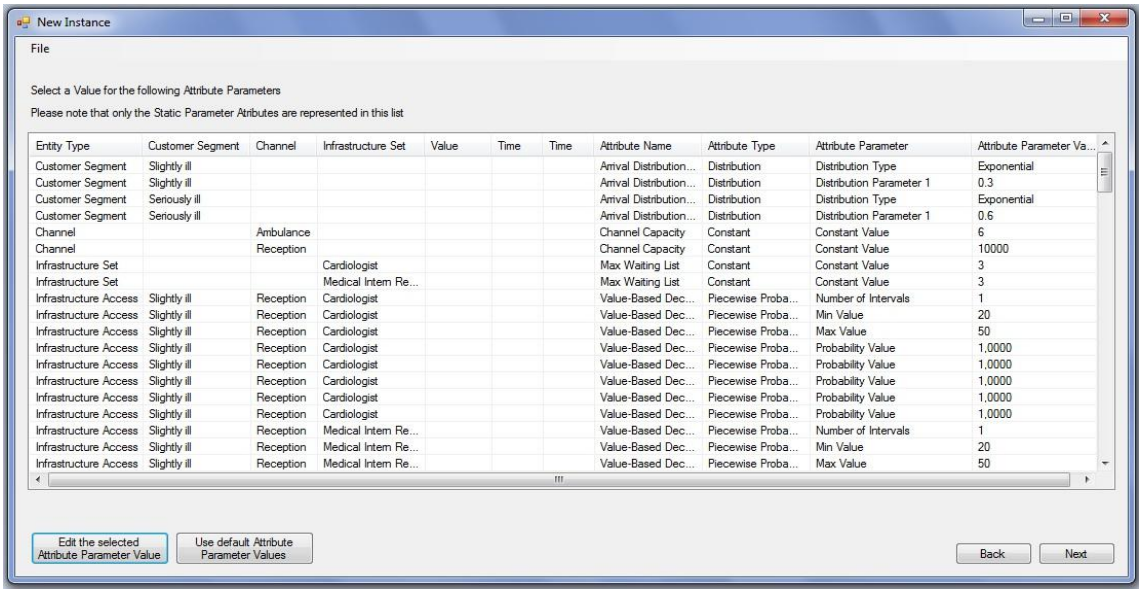


Figura 6-09. Ejemplo Sector Sanitario: Asignación de valores a Atributos estáticos

Los valores asignados a los atributos se incluyen en el Anexo A del proyecto, pero algunos valores relevantes son por ejemplo que los pacientes leves llegan al sistema siguiendo una distribución exponencial de media 0.3 horas, es decir, cada 18 minutos; los pacientes graves por su parte llegan cada 36 minutos (0.6 horas) de media. También destacar que la simulación se hace con una duranción de 6 horas, entre T Start igual a 0 y T End 6.

La aplicación ahora requiere la definición de los estados y cambios de estado. La primera parte del árbol de estados está fijada por el modelo y fue presentada en la figura 3-9 del capítulo 3. En cuanto a la parte del árbol definida por el analista, la figura 6-10 presenta tanto la secuencia de estados que puede tener lugar, como las condiciones que se deben cumplir para que se de cada cambio de estado o evento. En la figura 6-11 se incluye el formulario de definición de estados y cambios de estado de la aplicación.

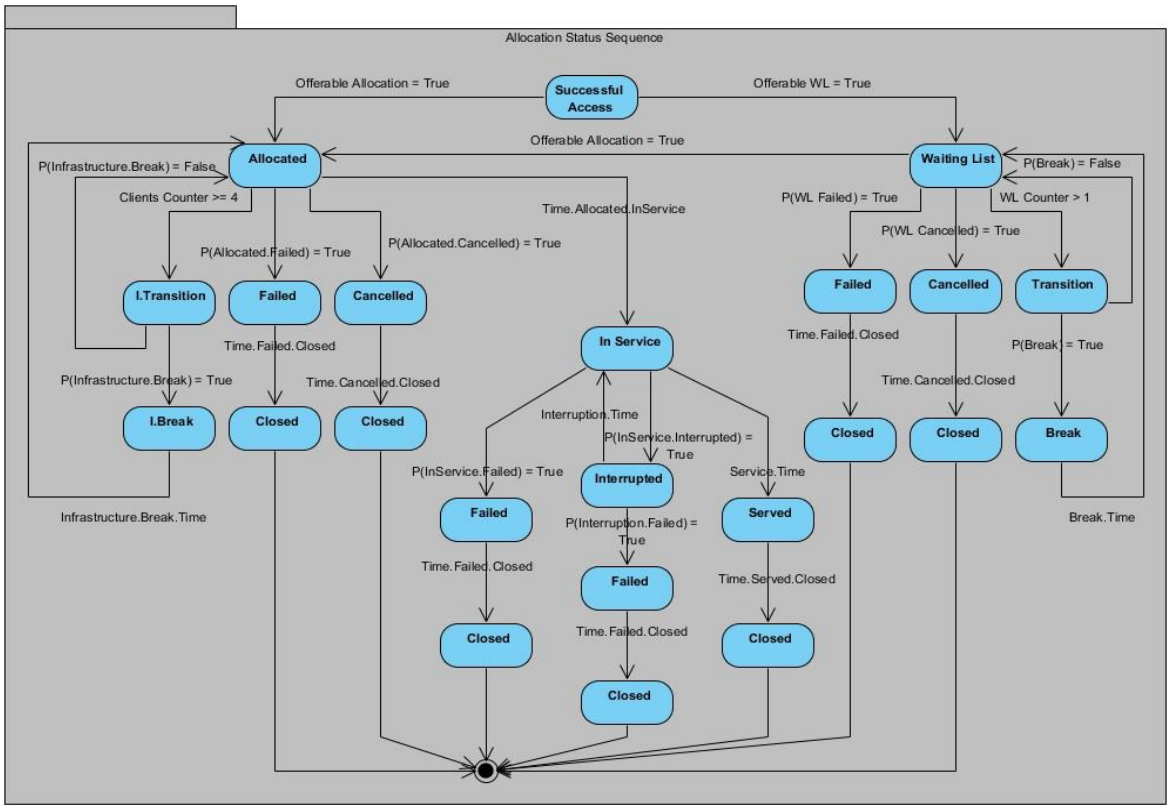


Figura 6-10. Ejemplo Sector Sanitario: Árbol de estados y caracterización de eventos

Una vez que un paciente, que ha accedido a través de un canal, haya sido asignado a un médico y pase al estado Successful Access, éste pasará a Allocated si hay médicos disponibles para atenderle o a lista de espera (Waiting List) en caso contrario. En caso de pasar a Allocated, la dinámica es la siguiente:

- Si el contador de clientes (Clients Counter) es mayor o igual a 4, es decir si han sido asignados al menos 4 pacientes desde el último descanso del médico, el paciente pasará al estado I.Transition, donde por probabilidad el médico puede tomarse un descanso y durar Infrastructure.Break.Time, o no tomar un descanso.
- Por otro lado, el paciente puede morir cuando está en el estado asignado y pasar a Failed. Tras un tiempo en el que el cuerpo es trasladado, terminará su proceso.
- La siguiente opción es que se marche voluntariamente, momento en el que pasará a Cancelled, y tras un tiempo en el que abandona el hospital acabará su proceso.

- La última opción, y la más común, es pasar a In Service (En servicio). En este punto puede morir el paciente y pasar a Failed, interrumpirse el servicio durante un tiempo (Interruption.Time) o morir durante la interrupción, o bien terminar con éxito el servicio y pasar a Served (Servido).

Si en cambio el paciente pasa a la lista de espera (Waiting List), la dinámica puede ser:

- Si hay más de un paciente en la lista de espera (WL Counter mayor que 1), el paciente pasará al estado Transition y habrá una probabilidad de dicho paciente se tome un descanso de tiempo Break.Time, como puede ser ir a la cafetería o a llamar por teléfono.
- El paciente también tiene una probabilidad de morir estando en lista de espera, y pasará a Failed. Del mismo modo, tiene una probabilidad de marcharse voluntariamente y pasar a Cancelled.
- Por último, si el paciente permanece en lista de espera, éste será asignado a un médico cuando quede disponible, y siempre se acuerdo al orden de prioridad definido por el usuario.

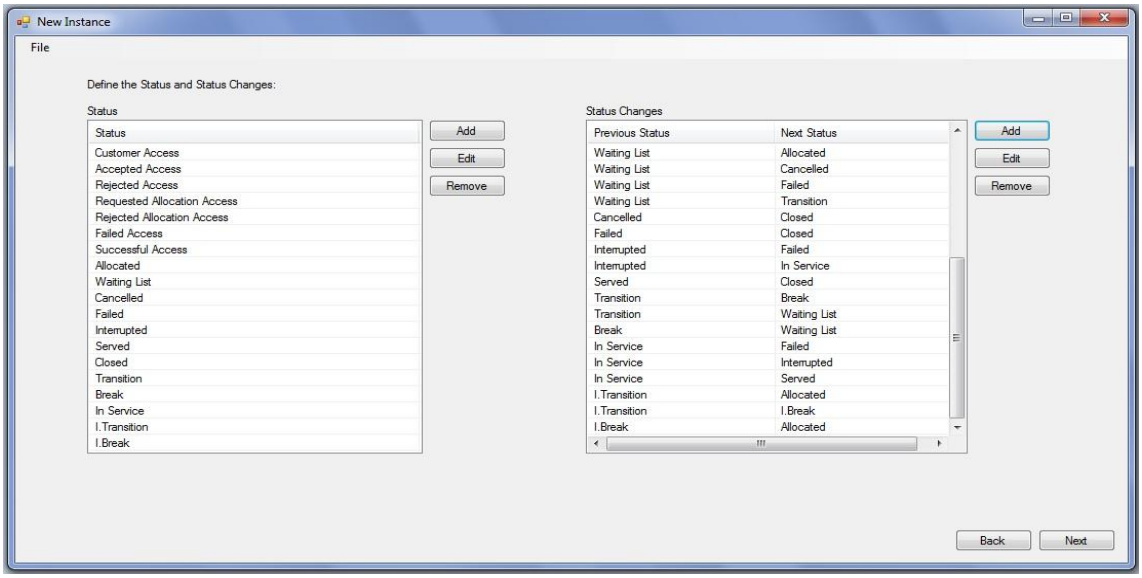
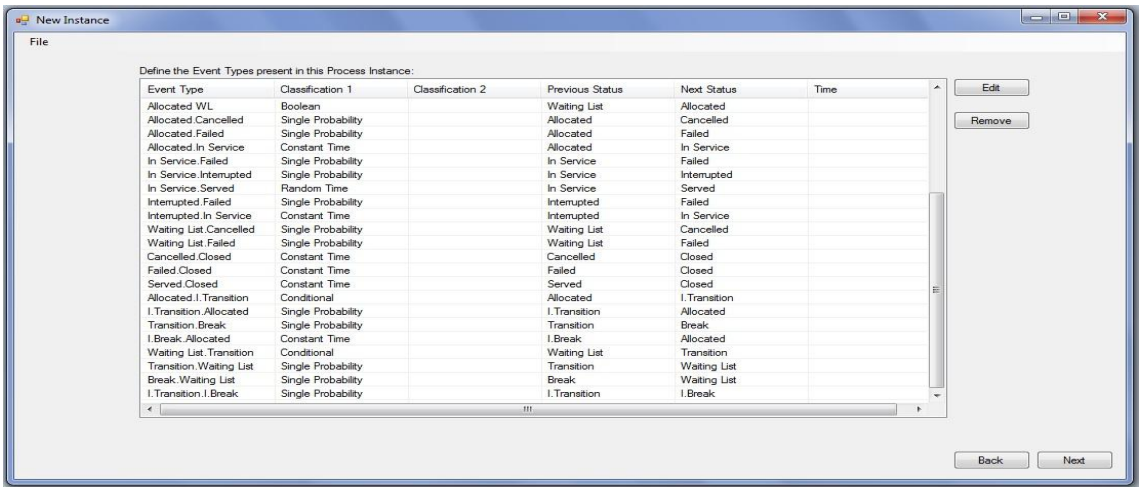


Figura 6-11. Ejemplo Sector Sanitario: Definición de Estados y Cambios de Estado

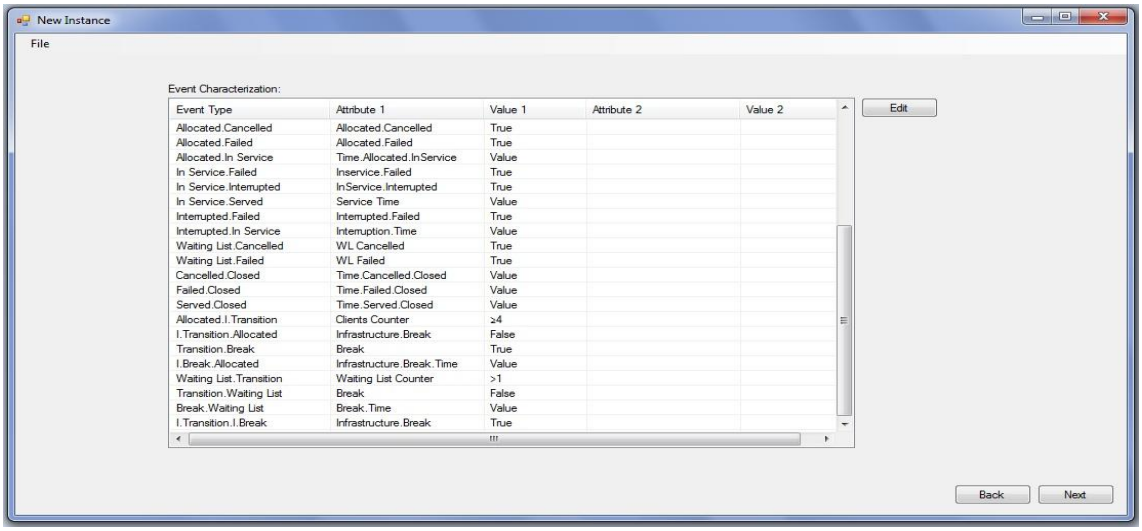
En el árbol de estados se pueden apreciar dos eventos que requieren de condiciones vinculadas a atributos de eventos dinámicos para su ocurrencia. Estos eventos, caracterizados por condiciones comparativas, muestran una gran funcionalidad que ofrece la herramienta, y en nuestro ejemplo el analista requiere de su uso. Se trata de las condiciones “Clients Counter ≥ 4”, y “WL Counter > 1”.

En la figura 6-12 se presenta la clasificación de los eventos, en base al tipo de atributos que los gobiernan. En la siguiente figura, figura 6-13, se muestra la caracterización de eventos, en donde se define tanto el atributo o atributos que posibilitan la ocurrencia del evento como el valor que éste debe adoptar. La caracterización de eventos se puede apreciar mejor en el árbol mostrado en la figura 6-10.



Event Type	Classification 1	Classification 2	Previous Status	Next Status	Time
Allocated.WL	Boolean		Waiting List	Allocated	
Allocated.Cancelled	Single Probability		Allocated	Cancelled	
Allocated.Failed	Single Probability		Allocated	Failed	
Allocated.In Service	Constant Time		Allocated	In Service	
In Service.Failed	Single Probability		In Service	Failed	
In Service.Interrupted	Single Probability		In Service	Interrupted	
In Service.Served	Random Time		In Service	Served	
Interrupted.Failed	Single Probability		Interrupted	Failed	
Interrupted.In Service	Constant Time		Interrupted	In Service	
Waiting List.Cancelled	Single Probability		Waiting List	Cancelled	
Waiting List.Failed	Single Probability		Waiting List	Failed	
Cancelled.Closed	Constant Time		Cancelled	Closed	
Failed.Closed	Constant Time		Failed	Closed	
Served.Closed	Constant Time		Served	Closed	
Allocated.I.Transition	Conditional		Allocated	I.Transition	
I.Transition.Allocated	Single Probability		I.Transition	Allocated	
Transition.Break	Single Probability		Transition	Break	
I.Break.Allocated	Constant Time		I.Break	Allocated	
Waiting List.Transition	Conditional		Waiting List	Transition	
Transition.Waiting List	Single Probability		Transition	Waiting List	
Break.Waiting List	Single Probability		Break	Waiting List	
I.Transition.I.Break	Single Probability		I.Transition	I.Break	

Figura 6-12. Ejemplo Sector Sanitario: Clasificación de eventos



Event Type	Attribute 1	Value 1	Attribute 2	Value 2
Allocated.Cancelled	Allocated.Cancelled	True		
Allocated.Failed	Allocated.Failed	True		
Allocated.In Service	Time.Allocated.InService	Value		
In Service.Failed	Inservice.Failed	True		
In Service.Interrupted	InService.Interrupted	True		
In Service.Served	Service Time	Value		
Interrupted.Failed	Interrupted.Failed	True		
Interrupted.In Service	Interruption.Time	Value		
Waiting List.Cancelled	WL Cancelled	True		
Waiting List.Failed	WL Failed	True		
Cancelled.Closed	Time.Cancelled.Closed	Value		
Failed.Closed	Time.Failed.Closed	Value		
Served.Closed	Time.Served.Closed	Value		
Allocated.I.Transition	Clients Counter	≥4		
I.Transition.Allocated	Infrastructure.Break	False		
Transition.Break	Break	True		
I.Break.Allocated	Infrastructure.Break.Time	Value		
Waiting List.Transition	Waiting List Counter	>1		
Transition.Waiting List	Break	False		
Break.Waiting List	Break.Time	Value		
I.Transition.I.Break	Infrastructure.Break	True		

Figura 6-13. Ejemplo Sector Sanitario: Caracterización de eventos

A continuación se define la función objetivo que determinará el valor más óptimo de las variables, es decir, el número óptimo de médicos cardiólogos e internos residentes para reducir costes sin reducir la calidad del servicio ofrecido. Para nuestro ejemplo sólo se requiere de una función objetivo del tipo maximización.

La función objetivo es la expresada a continuación, y consta de cuatro términos básicos:

$$F = \sum_{i=1}^{N} Value_i - 8000 \cdot d - \sum_{j=1}^{M}\sum_{k=1}^{L} c_j^w \cdot t_{kj}^w - Personnel\ Cost$$

- El primer término pretende potenciar la atención más adecuada del máximo número de pacientes. Para ello, se cuantifica de manera positiva el beneficio social que supone cada cliente que ha sido satisfactoriamente atendido. Se trata de un sumatorio de todos los pacientes atendidos con éxito (In Service – Served), representados por el valor social que conllevan (Value del elemento asignación).
- El segundo término está enfocado a penalizar duramente la muerte de pacientes mientras se encuentran en lista de espera, esperando a ser atendidos (Waiting List – Failed). Se penaliza con un valor de 8000 (c^d) cada paciente de cualquier tipo que muere mientras está en la lista de espera. Penaliza la falta de médicos.

- El tercer término también está enfocado a penalizar, aunque en menor medida, la falta de médicos. El término penaliza el tiempo que pasa cada paciente en la lista de espera (Waiting List). Para los pacientes Seriously ill (Seriamente enfermos), el coste de tiempo de espera por unidad de tiempo es 500 (c_2^w), y en el caso de pacientes leves (Slightly ill) es considerado 100 (c_1^w).
- El cuarto término está dedicado a valorar los costes de personal, los cuales queremos reducir y por ese motivo se valoran de forma negativa. Se considera un coste por unidad de tiempo (Por hora) de 200 para cada cardiólogo y de 10 para cada médico interno residente.

En la figura 6-14 se muestra el formulario correspondiente a la definición de la función objetivo. En las siguientes figuras (figuras 6-15, 6-16, y 6-17) se muestran las ventanas de consulta de los diferentes componentes dinámicos que conforman la función objetivo. En nuestro caso, tenemos 4 componentes de atributo, 2 componentes de estado, y 3 de cambio de estado.

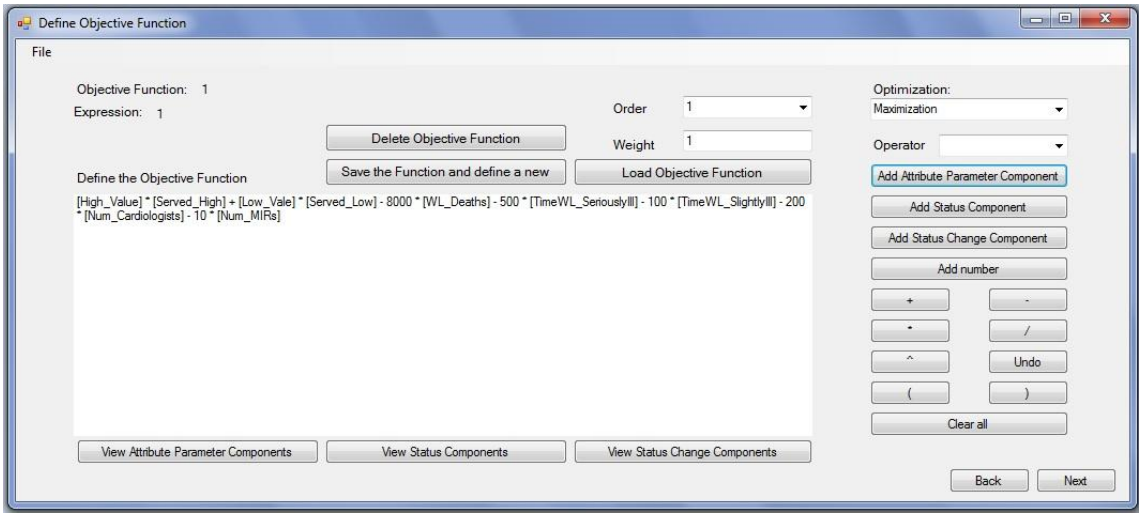


Figura 6-14. Ejemplo Sector Sanitario: Función Objetivo

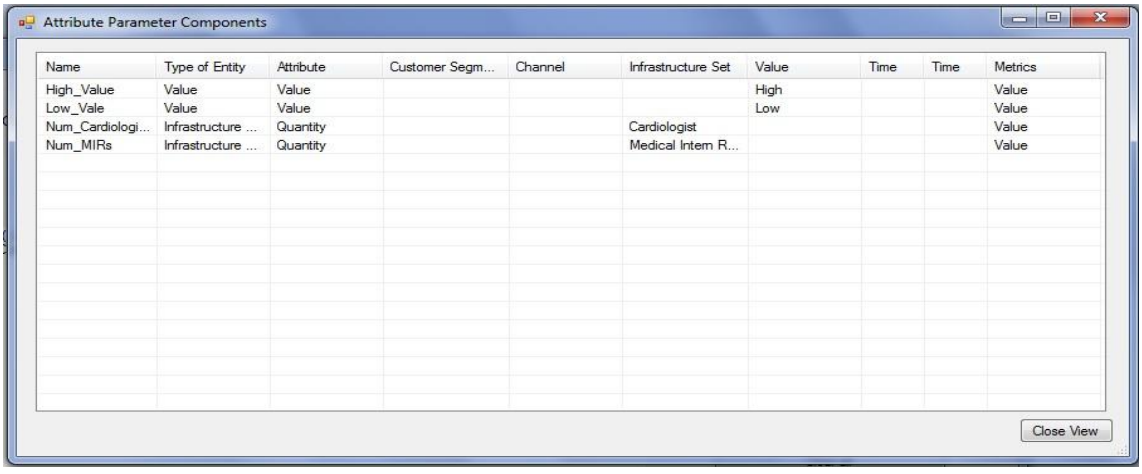


Figura 6-15. Ejemplo Sector Sanitario: Componentes de atributo

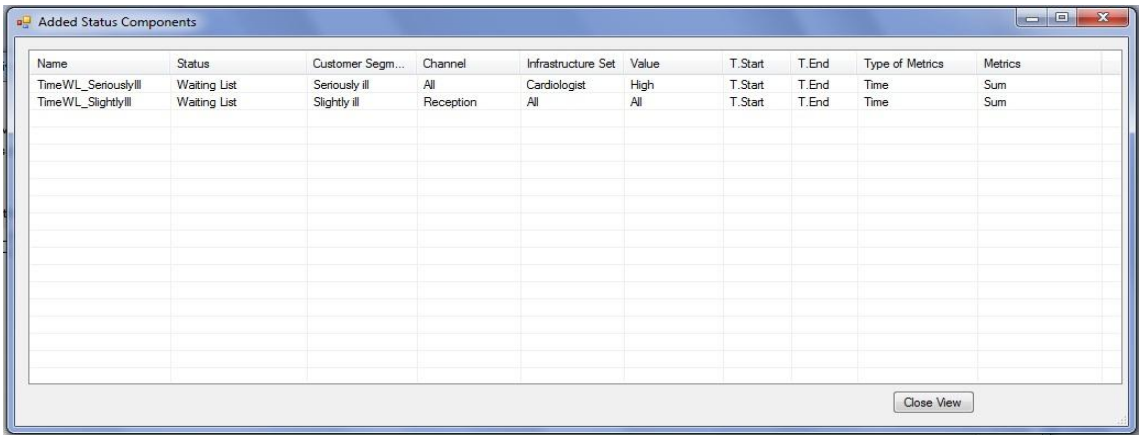


Figura 6-16. Ejemplo Sector Sanitario: Componentes de estado

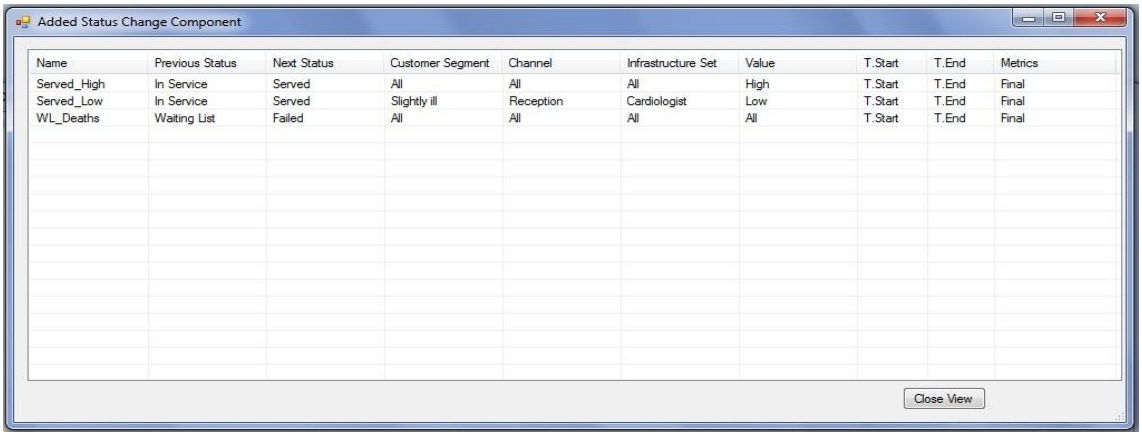


Figura 6-17. Ejemplo Sector Sanitario: Componentes de cambio de estado

Dentro de los atributos dinámicos definidos, se definen como atributos de evento dinámicos los fijados por el modelo (Infrastructure Availability, Channel Availability, Clock.Value), “Clients Counter”, y “WL Counter”. La caracterización de estos atributos se muestra en la figura 6-18. Destacar que el contador de clientes aumentará cada vez que un paciente pasa a ser asignado (Estado Allocated), y pasará a valor 0 cuando el médico se toma un descanso (I.Break).

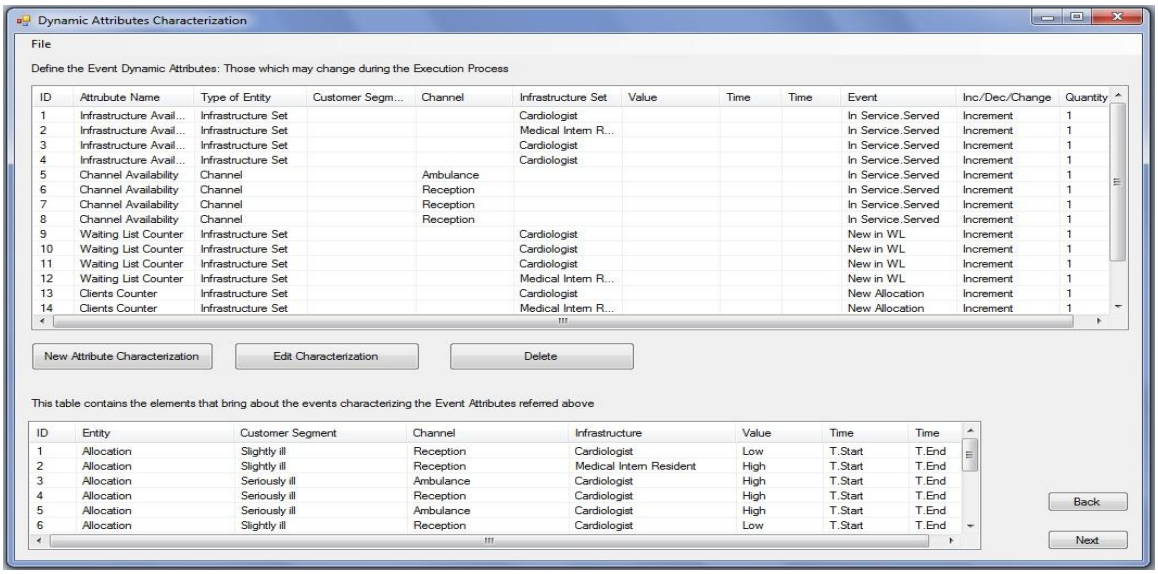


Figura 6-18. Ejemplo Sector Sanitario: Caracterización de atributos de evento dinámico

6.2 UNIDAD DE CORDIOLOGÍA DE URGENCIAS: DEFINICIÓN DEL CONJUNTO DE EJECUCIONES Y SIMULACIÓN

En este apartado, para terminar la tarea de definición por parte del analista del hospital, se trata la definición del nivel de conjunto de ejecuciones, y posteriormente se explica la simulación del proceso que realizaría la aplicación. Como estas dos partes no han podido ser implementadas en la aplicación por motivos de alcance del proyecto, se ha realizado una simulación completa en Microsoft Excel, de la misma manera que sería realizada por la aplicación, sirviendo de demostración y prueba de la robustez y validez de los modelos conceptuales y de datos.

En la definición del conjunto de ejecuciones, el analista determina que el número de ejecuciones/simulaciones a realizar debe ser de 20, y que las variables del problema y los valores iniciales de los atributos de evento dinámico clients Counter y WL Counter son:

Variables del Problema	Quantity	
	Cardiologist	Medical Intern Resident
	1	2

Valores iniciales de Atributos de evento dinámicos	

Clients Counter	
Cardiologist	Medical Intern Resident
0	0

WL Counter	
Cardiologist	Medical Intern Resident
0	0

Figura 6-19. Ejemplo Sector Sanitario: Valor de variables y valor inicial de atributos de evento dinámicos

En este punto, e analista ha definido todos los aspectos y propiedades del proceso de negocio, y el siguiente paso es la simulación. Se ha realizado una simulación completa en Excel, paso a paso y siguiendo los mismos procedimientos que debería hacer la aplicación. Por motivos de alcance no se ha podido programar la aplicación, pero su funcionamiento ha sido simulado, resultando en la lista de eventos añadida en el Anexo B del proyecto. A continuación se presentan los resultados de la evaluación de la función objetivo tras la simulación.

Objective Function
-7982

Status Change

InService - Served	
High	Low
20	1

Status Change

WL - Failed
All
1

Status (Medida de tiempo)

WL	
Seriously ill	Slightly ill
1,55	0,07

ID_Access	T.Start	T.End	Residence Time	
11	3,14	3,4	0,26	1,55
12	3,29	4,02	0,73	
19	4,18	4,38	0,2	
22	4,4	4,76	0,36	
20	4,41	4,48	0,07	0,07

Figura 6-20. Ejemplo Sector Sanitario: Evaluación Función Objetivo

Puesto que no es posible mostrar toda la simulación realizada en Excel, vamos a resaltar alguno de los puntos del proceso. En el momento de comenzar la ejecución (ClockTime = 0), en la tabla Execution Event List se cargan los eventos de tiempo y se programa la llegada del primer cliente Slightly ill y el primer cliente Seriously ill, y la caracterización de dichos eventos se carga en la tabla Execution Event characterization (Figura 6-21):

Execution Event List

ID_ProcessM	ID_Process	ExecutionSet	ID_ProcessEx	ID_Access	ID_EventType	Nº Event	_PreviousStat	D_NextStatu	Event Time	Occurrence Time
1	1	1	1	0	1	1			0	0
1	1	1	1	0	2	2			6	0
1	1	1	1	1	3	3	1	2	0,16	0
1	1	1	1	2	3	4	1	2	0,99	0

Execution Event Characterization

ID_ProcessN	ProcessInsta	ExecutionSet	ID_ProcessE	ID_Access	ID_EventType	D_IAM_Entit	_IAM_Eleme	ID_Attribute	NºEvent	Value_
1	1	1	1	0	1	6	1	1	1	0
1	1	1	1	0	2	6	2	1	1	6
1	1	1	1	1	3	1	1	1	3	0,16
1	1	1	1	2	3	1	2	1	4	0,99

Figura 6-21. Ejemplo Sector Sanitario: Tablas de base de datos en inicialización de simulación

Además, en el momento de cargar de la simulación, también se cargan los valores iniciales de los atributos de evento dinámicos en las tablas InfrastructureExecutionStatus (Disponibilidad de Médicos incorporando Lista de espera), ChannelExecutionStatus (Disponibilidad de canales), y EventAttributeExecutionStatus (Clients Counter, WL Counter, y Clock.Time), entre los que está el reloj de la ejecución, Clock.Time (Figura 6-22):

Event Attribute Execution Status

ID_ProcessN	ID_Process	ExecutionSet	ID_ProcessE	ID_Access	D_EventType	D_IAM_Entity	AM_Element	ID_Attribute	NºEvent	Value_	
1	1	1	1	0	1	3	1	3	1	0	WL Counter
1	1	1	1	0	1	3	1	4	1	0	Clients Counter
1	1	1	1	0	1	3	2	3	1	0	WL Counter
1	1	1	1	0	1	3	2	4	1	0	Clients Counter
1	1	1	1	0	1	6	3	2	1	0	Clock.Time

Infrastructure Execution Status

ID_ProcessM	ID_Process	ExecutionSet	ID_ProcessEx	ID_Access	ID_EventType	NºEvent	ID_IAM_Elem	Availability_I	
1	1	1	1	0	1	1	1	4	Quantity (1) + Waiting List (3) de Cardiologist
1	1	1	1	0	1	1	2	5	Quantity (2) + Waiting List (3) de Medical Intern

Channel Execution Status

ID_ProcessM	ID_Process	ExecutionSet	ID_ProcessEx	ID_Access	ID_EventType	NºEvent	ID_IAM_Elem	Availability_C	
1	1	1	1	0	1	1	1	6	Capacity de Ambulance
1	1	1	1	0	1	1	2	1000	Capacity de Reception

Figura 6-22. Ejemplo Sector Sanitario: Tablas de base de datos en inicialización de simulación

Un ejemplo de llegada de un cliente se muestra a continuación, y se recoge en la lista de eventos. Destacar que cada vez que llega un cliente de un tipo se programa la llegada del siguiente cliente de ese tipo.

Time	Eventos	
0,16	Llega un customer tipo 1 slighttly ill De acuerdo a Priority, es asignado ID_Allocation = 2 Quantity de Medical Intern se reduce en 1. También se reduce channel, reception. El Customer pasará a In service en Time.Allocated.InService (0,03)	E1

Como se refleja en la lista de eventos, se reduce un Medical intern, ya que es asignado, y se reduce 1 de la capacidad del canal recepción (Figura 6-23):

Infrastructure Execution Status

ID_ProcessM	ID_Process	ExecutionSet	ID_ProcessEx	ID_Access	ID_EventType	NºEvent	ID_IAM_Elem	Availability_I
1	1	1	1	1	10	9	2	4

Channel Execution Status

ID_ProcessM	ID_Process	ExecutionSet	ID_ProcessEx	ID_Access	ID_EventType	NºEvent	ID_IAM_Elem	Availability_C
1	1	1	1	1	10	9	2	999

Figura 6-23. Ejemplo Sector Sanitario: Cambio en el valor de atributos de evento debido al evento 1

Con respecto a los otros atributos de evento dinámico, el reloj de la ejecución se actualiza, y el contador de clientes asignados (Clients Counter) aumenta a uno (Figura 6-24).

Event Attribute Execution Status

ID_ProcessN	ID_ProcessIn	xecutionSet	ID_ProcessE	ID_Access	ID_EventType	ID_IAM_Entity	AM_Element	ID_Attribute	NºEvent	Value_	
1	1	1	1	1	3	6	3	2	3	0,16	Clock.Value
1	1	1	1	1	10	3	2	4	9	1	
1	1	1	1	1	10	3	1	4	9	1	

Figura 6-24. Ejemplo Sector Sanitario: Cambio en el valor de atributos de evento debido al evento 1

Por último, y para finalizar el ejemplo de la simulación, se presentan los registros correspondientes al evento número 54, en el tiempo 3,83 de la simulación.

Time	Eventos	
3,83	Access 16, que había muerto estando en lista de espera, termina su paso por el proceso Llega un customer tipo 1 slightly ill (Access 17) Solicita Medical Intern y pasa a Waiting List Al estar en WL, se toma un descanso y se va a la cafetería. Pasará a WL.	E54

Aquí, Access 16 termina su paso por el proceso tras su muerte. Llega un cliente Slightly ill (Access 17), y por prioridad, probabilidad, y disponibilidad de infraestructuras, accede por recepción y requiere de un médico interno. Pasa por los estados de inicio hasta llegar a la lista de espera (WL Counter aumenta en 1), porque no hay médicos internos disponibles. Una vez en lista de espera se toma un descanso (Estado Break) y va a la cafetería durante 0,27 horas (WL Counter disminuye en 1), tiempo tras el que volverá a la lista de espera. También se programa la llegada del siguiente cliente Slightly ill (Access 18).

Execution Event List

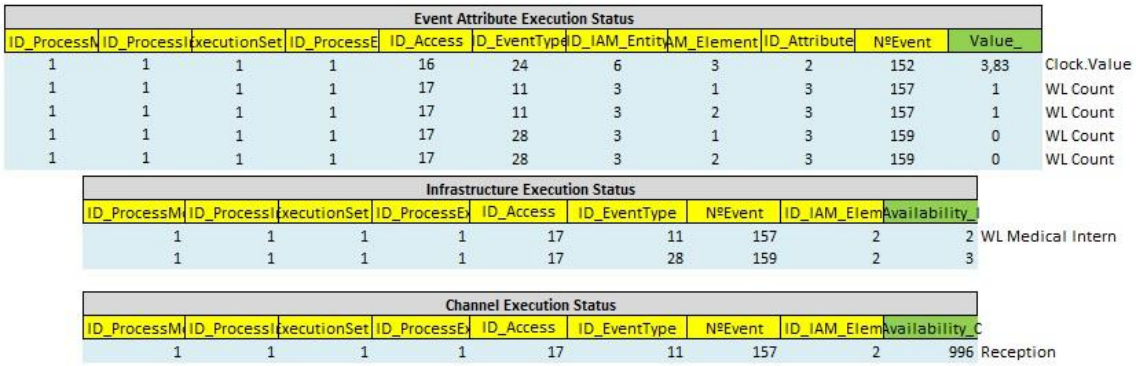
ID_ProcessM	ID_ProcessIn	xecutionSet	ID_ProcessEx	ID_Access	ID_EventType	Nº Event	_PreviousStat	D_NextStatu	Event Time	Occurrence Event
1	1	1	1	16	24	152	12	15	3,83	3,75
1	1	1	1	18	3	153	1	2	4,17	3,83
1	1	1	1	17	4	154	2	3	3,83	3,83
1	1	1	1	17	6	155	3	5	3,83	3,83
1	1	1	1	17	9	156	5	8	3,83	3,83
1	1	1	1	17	11	157	8	10	3,83	3,83
1	1	1	1	17	30	158	10	16	3,83	3,83
1	1	1	1	17	28	159	16	17	3,83	3,83
1	1	1	1	17	32	160	17	10	4,1	3,83

Execution Event Characterization

ID_ProcessN	ProcessInsta	xecutionSet	ID_ProcessE	ID_Access	ID_EventType	D_IAM_Entit	_IAM_Eleme	ID_Attribute	NºEvent	Value_	Atribute_name
1	1	1	1	16	24	4	3	6	152	0,08	Time.Failed.Closed
1	1	1	1	18	3	1	1	1	153	0,34	Arrival Distribution
1	1	1	1	17	4	7	2	1	154	True	Offerable
1	1	1	1	17	6	7	2	2	155	True	Requested
1	1	1	1	17	9	4	2	2	156	False	Failed Access Probability
1	1	1	1	17	11	7	2	4	157	True	Offerable WL
1	1	1	1	17	30	3	2	3	158	1	WL Count
1	1	1	1	17	28	4	2	10	159	True	Break
1	1	1	1	17	32	4	2	11	160	0,27	Break,Time

Execution Event List

ID_ProcessM	ID_ProcessIn	xecutionSet	ID_ProcessEx	ID_Access	ID_EventType	Nº Event	_PreviousStat	D_NextStatu	Event Time	Occurrence Event
1	1	1	1	16	24	152	12	15	3,83	3,75
1	1	1	1	18	3	153	1	2	4,17	3,83
1	1	1	1	17	4	154	2	3	3,83	3,83
1	1	1	1	17	6	155	3	5	3,83	3,83
1	1	1	1	17	9	156	5	8	3,83	3,83
1	1	1	1	17	11	157	8	10	3,83	3,83
1	1	1	1	17	30	158	10	16	3,83	3,83
1	1	1	1	17	28	159	16	17	3,83	3,83
1	1	1	1	17	32	160	17	10	4,1	3,83



Event Attribute Execution Status

ID_ProcessN	ID_ProcessI	ExecutionSet	ID_ProcessE	ID_Access	ID_EventType	ID_IAM_Entity	AM_Element	ID_Attribute	NºEvent	Value_	
1	1	1	1	16	24	6	3	2	152	3,83	Clock.Value
1	1	1	1	17	11	3	1	3	157	1	WL Count
1	1	1	1	17	11	3	2	3	157	1	WL Count
1	1	1	1	17	28	3	1	3	159	0	WL Count
1	1	1	1	17	28	3	2	3	159	0	WL Count

Infrastructure Execution Status

ID_ProcessM	ID_ProcessI	ExecutionSet	ID_ProcessEx	ID_Access	ID_EventType	NºEvent	ID_IAM_Elem	Availability_I	
1	1	1	1	17	11	157	2	2	WL Medical Intern
1	1	1	1	17	28	159	2	3	

Channel Execution Status

ID_ProcessM	ID_ProcessI	ExecutionSet	ID_ProcessEx	ID_Access	ID_EventType	NºEvent	ID_IAM_Elem	Availability_C	
1	1	1	1	17	11	157	2	996	Reception

Figura 6-24. Ejemplo Sector Sanitario: Tablas de base de datos en evento 54 de la simulación

CAPÍTULO 7

CONCLUSIONES

Una vez terminado el desarrollo e implementación del actual proyecto, es hora de hacer balance de los logros, la culminación de objetivos y expectativas, y de exponer posibles mejoras futuras de la herramienta, y en la misma línea de investigación. En una primera parte se analiza el resultado obtenido, para en una segunda parte citar los posibles desarrollos futuros que permitan cerrar y mejorar la herramienta.

7.1 CONCLUSIONES

En el punto de partida del actual proyecto, se disponía de modelos conceptuales, modelo de datos, e implementación de la aplicación hasta el nivel de instancia de proceso. El objetivo inicial, de acuerdo al objetivo principal (OP) y a los dos objetivos complementarios (OC1 y OC2) explicados en el apartado 1.2 de objetivos del proyecto; se pretendía desarrollar los modelos conceptuales, modelo de datos, e implementación de la herramienta para consolidar el nivel de instancia y posteriormente desarrollar el nivel de ejecución, es decir, la simulación del proceso definido. Siempre teniendo en cuenta que la meta es desarrollar una herramienta flexible, rápida, e intuitiva para la resolución y optimización de problemas de asignación de infraestructuras a clientes, pertenecientes a un segmento, que acceden por distintos canales para reservar una infraestructura.

Podemos resumir que acorde al objetivo principal (OP), la meta marcada era:

- Adaptación del modelo estático.
- Crear y desarrollar un nuevo modelo dinámico.
- Crear una función Multiobjetivo.
- Crear y desarrollar la capa Execution Set.

En relación al objetivo complementario 1 (OC1), la meta marcada inicialmente era el diseño e implantación de un sistema de tratamiento de plantillas, así como la corrección y mejora de la aplicación en toda su extensión.

En el segundo objetivo complementario (OC2), la principal meta era la creación de los modelos conceptuales y de base de datos del nivel de ejecución, así como la simulación manual de un ejemplo.

Con este escenario, se comenzó a depurar y mejorar los modelos conceptuales pertenecientes a los módulos de estado y de función objetivo respectivamente, pertenecientes a la instancia de proceso. Al tratarse de un proyecto de investigación, la necesidad de cambios en los modelos de partida nos llevó a replantear todos los modelos conceptuales desde el nivel de metamodelo. Por esto, debido a necesidades de los modelos conceptuales, el proyecto se reorientó y se centró en el desarrollo de los nuevos modelos conceptuales desde el primer nivel. A su vez, estos cambios en los modelos conceptuales, por tratarse de la espina dorsal del proyecto, nos llevo a cambiar tanto el modelo de datos (Base de datos) como la implementación de la aplicación en muchos sentidos.

Desde el punto de vista de las distintas fases que han de producirse en todo proyecto de investigación, en el actual proyecto se ha realizado un nuevo "loop" (Ciclo) de mejora de todo lo que se había desarrollado hasta el momento (Modelos conceptuales, de datos, y aplicación). Como parte fundamental, y que explicaremos a

continuación, se han añadido nuevas aportaciones de acuerdo a los objetivos, como una nueva orientación en el módulo de la función objetivo y de atributos, el nivel de conjunto de ejecuciones (Nuevo), o el nivel de simulación; así como la implantación de un sistema de tratamiento de plantillas de los modelos y las instancias de procesos. En todo momento se ha tenido en cuenta la finalidad de ofrecer una herramienta lo más flexible posible, y que contenga funcionalidades que permitan modelar un problema de forma lo más cercano posible a la realidad. Puesto que hemos tenido que rehacer los modelos conceptuales, se ha procurado aprovecharlo para añadir nuevas opciones y posibilidades de desarrollo futuro, para aumentar la potencia de la aplicación.

Con respecto a las aportaciones del proyecto, cabe destacar varias. Comenzando por los niveles de metamodelo y modelo de proceso de negocio, se han reconstruido los modelos conceptuales, base de datos, y en consecuencia se modificado la programación de la aplicación, implantado una programación más modular y eficiente, manteniendo los formularios del proyecto anterior pero añadiendo nuevas posibilidades. Entre estas nuevas funcionalidades, destaca el diseño e implantación de un sistema de tratamiento de plantillas, que permite tanto guardar plantillas de modelos, como cargarlas. Esto permite aumentar la rapidez en la definición de modelos, así como facilitar al usuario en gran medida el proceso de definición. Este sistema de plantillas conlleva tanto su implantación en la aplicación, como en la base de datos. A su vez, se han corregido muchos problemas en la programación para aumentar la robustez de la aplicación.

En el nivel de instancia, los modelos de atributos, estados, y función objetivo han sufrido profundos cambios, lo que lleva a los cambios correspondientes en la base de datos y en la aplicación. En este sentido, se han desarrollado modelos conceptuales robustos, corrigiendo deficiencias existentes anteriormente. Destaca el nuevo modelo de atributos, en el que se han incorporado los atributos dinámicos (Pueden cambiar durante la ejecución), y que permiten incluir eventos, y términos de la función objetivo que acerquen más el modelo a la realidad del proceso de negocio. Estos atributos dinámicos son una de las razones por las que se ha añadido un nivel jerárquico más. En cuanto al modelo de estados y eventos, también ha sido reconfigurado y se le ha dotado de robustez. Se ha mejorado las posibilidades de caracterización de eventos, pudiéndose añadir incluso condiciones comparativas para la ocurrencia de eventos. También destaca el nuevo enfoque de la función objetivo, en la que se pueden definir multitud de ellas en una misma instancia de acuerdo a una prioridad y un peso, y funciones objetivo de tipo lógicas (Restricciones). La mejora en el modelo de atributos también afecta positivamente a la definición de las funciones objetivo.

En la instancia, ha sido necesario fijar una serie de atributos, así como la parte inicial del árbol de estados, para garantizar la coherencia de los modelos y que las ejecuciones de los modelos se realicen acorde a éstos, siempre siguiendo una secuencia lógica y realista. La instancia también se ha dotado de un sistema de tratamiento de plantillas, mediante el que se pueden guardar y cargar plantillas de instancias. De nuevo, este sistema de plantillas está orientado a aumentar la rapidez en la definición de instancias, y reducir el tiempo de familiarización del usuario con la aplicación. Este sistema se ha implantado a través de botones y opciones, que permiten al usuario de forma intuitiva y sencilla manejarlo, y a través de su implantación en la base de datos. Además, en el nivel de instancia de la aplicación se han añadido formularios nuevos y se ha modificado toda la programación.

El nivel de conjunto de ejecuciones (Execution Set) ha sido implantado en el desarrollo del proyecto. Se trata de un nuevo nivel jerárquico, que surgió por necesidades del modelo y como consecuencia del proceso de mejora que ha sufrido la línea de investigación, y que encaja perfectamente. Por ello, tanto el diseño de los modelos conceptuales, como el diseño de la base de datos son totalmente novedosos.

El último nivel, el nivel de ejecución (Execution), también es totalmente novedoso. Al igual que en el caso anterior, los modelos conceptuales y la base de datos se han construido de cero y en línea con los nuevos modelos de base. Estos últimos dos niveles no han podido ser implementados en la aplicación por motivos de alcance: sin embargo, se ha realizado un ejemplo real con todo nivel de detalle en Microsoft Excel, simulando el proceso de la misma manera que haría la aplicación y rellenando todas las tablas de la base de datos. Por este motivo, aunque no estén implantados están totalmente definidos, probados, y sólo falta su traducción al lenguaje de programación oportuno.

Resumido el proceso global y las aportaciones, cabe destacar que, aunque no se aprecie en la documentación del presente proyecto, se ha realizado una costosa y duradera tarea de programación y corrección de errores para conseguir una aplicación más robusta. En cuanto a los objetivos y expectativas del proyecto, hemos conseguido desarrollar nuevos modelos conceptuales más robustos, que han dado lugar a una aplicación flexible, que permite a los usuarios definir de forma rápida, intuitiva, y guiada a través de los formularios, problemas de asignación de infraestructuras.

Se ha desarrollado una aplicación, basada en los modelos, con la cual si el usuario adquiere un alto grado de dominio de las posibilidades que ésta ofrece, se puede optimizar a priori cualquier problema de asignación de infraestructuras de forma eficiente y con resultados muy cercanos a la realidad. Si el usuario es capaz de exprimir las posibilidades que ofrece la aplicación, en especial los atributos (Dinámicos), la caracterización de eventos y la nueva función objetivo (Función objetivo múltiple), la herramienta adquiere una gran potencia para la simulación y resolución de los problemas objetivo.

Respecto al uso de las herramientas empleadas para los tres bloques fundamentales del proyecto, a pesar de que ha sido necesario un aprendizaje desde cero, ha resultado muy satisfactorio. Se trata de herramientas muy útiles y versátiles, de las cuales hemos obtenido el rendimiento esperado: UML como herramienta unificada para el modelado de los modelos conceptuales, IDEF1X como herramienta intuitiva para el modelo de datos, y Visual Basic que dispone de interfaz gráfica con el usuario como lenguaje de programación para la implementación de la aplicación software. Además, se ha empleado Microsoft Access para la implementación de la base de datos.

Por todo lo mencionado anteriormente, el resultado del proyecto desde el punto de vista de la consecución de objetivos es muy satisfactorio. Después de un largo proceso de desarrollo e investigación, se ha conseguido demostrar a través de la implementación de la aplicación la validez y robustez de los modelos genéricos creados. La aplicación, tal y como se requería, permite definir de forma muy flexible y rápida, en la medida de lo posible, problemas de asignación de infraestructuras muy diferentes entre sí. Además, se ha realizado una simulación completa de un ejemplo, llegando hasta el objetivo final que es la simulación y optimización de los procesos, demostrando también aquí la consistencia de los modelos.

7.2 DESARROLLOS FUTUROS

La línea de desarrollo futura más clara, y necesaria para la completa finalización de esta línea de investigación, es la implementación de la simulación de los procesos en la aplicación. Esta simulación ha de ser en base a los modelos conceptuales y a los conceptos explicados en el nivel de simulación (Apartado 3.6.2), y basado en el ejemplo de uso de la herramienta, donde la simulación de ha desarrollado en Excel. Es aconsejable que se implante la simulación en la plataforma Visual Basic, para completar la aplicación en este lenguaje y entorno de programación; no obstante existen otras herramientas como Arena que pueden ser útiles para la programación de las ejecuciones.

Otro posible desarrollo futuro es la mejora del formulario de definición de cambios de estado. En el formulario actual se presentan los cambios de estado en forma de lista, y ésta lista podría ser sustituida por una interfaz gráfica que mostrase la secuencia de estados en forma de árbol. La presentación sería de manera similar a los árboles de estado en UML, y que se han mostrado durante el proyecto. Se podría incluso aprovechar este árbol para caracterizar los eventos de cambio de estado sobre las líneas que unen los estados, de igual manera que en la figura 6-10 (Capítulo 6). Esto ayudaría a hacer la aplicación más intuitiva.

Existen dos posibles desarrollos futuros que se han tenido en cuenta en el los nuevos modelos conceptuales, abriendo sendas posibilidades para el futuro. La primera está relacionada con la caracterización de eventos. Como hemos explicado, existe la posibilidad de caracterizar eventos mediante relaciones condicionales con atributos dinámicos, por ejemplo número clientes en lista de espera (Atributo dinámico) > 5. Para el desarrollo futuro se podría caracterizar un evento relacionándolo con una expresión matemática (Mathematical Expression). Esto sería un desarrollo muy factible, ya que los modelos están pensados para su implantación. Por un lado tenemos expresiones matemáticas, y por otro la posibilidad de caracterizar eventos con relaciones condicionales. Desde el punto de vista de la base de datos, ésta también está pensada para incluir el desarrollo de manera sencilla. Podría valer con añadir un campo llamado ID_Expression en la tabla “Event Characterization Definition”. No se ha implementado por motivos de alcance, pero se trabajó en dejar el modelo y base de datos preparados para este fin.

De la misma manera, se trabajó para adaptar tanto los modelos como la base de datos para el siguiente desarrollo futuro. Se trata de vincular el valor de un parámetro de atributo a una expresión matemática. Por lo tanto, el valor del parámetro de atributo será evaluado de acuerdo a dicha expresión matemática a la hora de usar el atributo. Un ejemplo podría ser un parámetro de atributo cuyo valor dependiera del tiempo de ejecución (Clock.Time) a través de una expresión. De nuevo, sería relacionar el campo ID_Expression con un parámetro de atributo. Estos atributos cuyo valor depende de una expresión matemática serían dinámicos.

Estos dos últimos desarrollos fueron planteados durante el proyecto, y se preparó para su implementación; no obstante, por motivos de alcance no se llegaron a implantar y podrían aportar mayor potencia a la aplicación.

CAPÍTULO 8

BIBLIOGRAFÍA

Bibliografía

1. AITKEN, Peter G. *Visual Basic 6: manual completo de programación.* 2ª Edición. Madrid: Paraninfo, 2002. 732 p. ISBN: 8428325456.

2. CARRASCO, Fernando José. "Herramienta informática para modelado flexible de problemas de asignación de infraestructuras". Tutor: Miguel Gutiérrez. Proyecto Fin de Carrera. Universidad Carlos III de Madrid, Departamento de Ingeniería Mecánica, 2011.

3. CHAVEZ MIRANDA, M.E; RUIZ JIMENEZ, A. Marco conceptual del Yield Management como técnica de gestión de la capacidad y la demanda en organizaciones de servicios. *Investigaciones Europeas de Dirección y Economía de la Empresa.* Vol. 11, Nº 1, 2005, pp. 143-163, ISSN: 1135-2523

4. *Curso de Access 2010*. Obtenido el 3 de Octubre de 2011 de AulaClic S.L., Access 2010: http://www.aulaclic.es/access-2010/index.htm

5. FOWLER, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language.* Third Edition. USA: Addison-Wesley, 2004. 175 p. ISBN: 978-0321193681.

6. GUTIÉRREZ, Miguel; DURÁN, Alfonso. *Generic Model Base Design for Decision Support Systems in Revenue Management: Applications to Hotel and Health Care Sectors*. 2011.

7. KELTON, W. David; LAW, Averill M. *Simulation Modeling and Analysis*. Third Edition. USA: McGraw-Hill, 2000. 760 p. ISBN: 0-07-116537-1.

8. SILVER, Mark S. *Systems That Support Decision Makers: Description and Analysis*. England: John Wiley & Sons, 1991. 254 p. ISBN: 0-471-91968-3.

9. SPRAGUE, Ralph H; CARLSON, Eric. *Building Effective Decision Support Systems*. 1ª Edición. EEUU: Prentice Hall, 1982. 304 p. ISBN: 978-0130862150.

10. SWARTZFAGER, Gene. *Visual Basic 6: programación orientada a objetos.* Madrid: Paraninfo, 1999. 454 P. ISBN: 8428325545

11. THOMSEN, Carsten. *Programación de Bases de Datos con Visual Basic .NET*. España: Inforbook's Ediciones, 2002. 479 p. ISBN: 84-95318-89-X.

ANEXOS

ANEXO A

Valores de los atributos estáticos en el ejemplo del capítulo 6, referente a la Unidad de cardiología de un hospital:

	Infrastructure Access		
ID	ID_Customer Seg.	ID_Channel	ID_Infrastructure
1	Slightly ill	Reception	Cardiologist
2	Slightly ill	Reception	Medical Intern Resident
3	Seriously ill	Ambulance	Cardiologist
4	Seriously ill	Reception	Cardiologist

	Allocation					
ID	ID_Customer	ID_Channel	ID_Infrastructure	Value	Start	End
1	Slightly ill	Reception	Cardiologist	Low	Ti	Tf
2	Slightly ill	Reception	Medical Intern Resident	High	Ti	Tf
3	Seriously ill	Ambulance	Cardiologist	High	Ti	Tf
4	Seriously ill	Reception	Cardiologist	High	Ti	Tf

Infrastructure Access	ID Infrastructure Access			
	1	2	3	4
Value-Based Decision				
Failed Access Probability	0	0	0	0
Service Time	0,15	0,2	0,3	0,3
Time.Allocated.InService	0,03	0,03	0,05	0,05
Time.Served.Closed	0,02	0,02	0,04	0,04
Time.Failed.Closed	0,08	0,08	0,08	0,08
Time.Cancelled.Closed	0,02	0,02	0,03	0,03
WL Failed	0,02	0,02	0,35	0,4
WL Cancelled	0,1	0,1	0	0
Break	0,3	0,3	0	0
Break.Time	0,15	0,15	0	0
Allocated.Cancelled	0	0	0	0
Allocated.Failed	0,02	0,02	0,1	0,15
Inservice.Failed	0,01	0,01	0,08	0,1
InService.Interrupted	0,2	0,2	0,05	0,05
Interruption.Failed	0,01	0,01	0,15	0,2
Interruption.Time	0,06	0,06	0,03	0,03
Infrastructure.Break	0,4	0,4	0	0
Infrastructure.Break.Time	0,05	0,05	0	0

Value-Based Decision	ID Infrastructure Access			
	1	2	3	4
N° Intervals	1	1	1	1
Max	50	50	50	50
Min	20	20	20	20
Probability	1	1	1	1

	Customer	
	Slightly ill	Seriously ill
rrival Distribution Dat	0,3	0,6

	Time	
	T.Start	T.End
time	0	6

	Channel	
	Ambulance	Reception
Channel Capacity	6	10000

	Value	
	High	Low
value	50	20

	Imnfrastructure Set	
	Cardiologist	Quantity
Quantity	1	2
Max Waiting List	3	3

Allocation	ID Allocation			
	1	2	3	4
Priority	2	1	2	1
Access Election Probability	1	0,9	1	0,6
Priority_WL	3	4	1	2

ANEXO B

A continuación se incluye la lista de eventos sucedidos en la simulación del ejemplo de la unidad de cardiología:

Time	Eventos	
0	Se cargan los tiempos en Event List (T.Start y T.End) Se cargan valores iniciales en Infrastructure Execution Status Se cargan valores iniciales en Channel Execution Status Se cargan valores iniciales en Event Attribute Execution Status y se inicializa el clock.Value Se programa la llegada del primer customer de cada tipo	E0
0,16	Llega un customer tipo 1 slighttly ill De acuerdo a Priority, es asignado ID_Allocation = 2 Quantity de Medical Intern se reduce en 1. También se reduce channel, reception. El Customer pasará a In service en Time.Allocated.InService (0,03)	E1
0,19	Access 1 pasa a In Service y pasa a interrupted Access 1 pasará de nuevo a In Service en Interruption Time (0,06)	E2
0,25	Access 1 pasa a In Service y pasará a Served tras service Time	E3
0,45	Access 1 pasa a Served y acabará su paso por el proceso en Time.Served.Closed	E4
0,47	Access 1 finaliza su paso por el proceso	E5
0,99	Llega un customer tipo 2 Seriously ill (Access 2) Access 2 entra en allocated y muere. Pasará a closed.	E6
1,07	Access 2 pasa a closed y termina su paso por el proceso.	E7
1,39	Llega un customer tipo 1 slighttly ill (Access 3) Pasa a Allocated con Medical Intern y pasará a In Service	E8
1,41	Llega otro customer tipo 1 Slightly ill (Access 4) El medical Intern no se toma un Break y el customer pasará a In Service	E9
1,42	Access 3 pasará a served en service Time (0,38)	E10
1,44	Access 4 pasa a In Service y fallece durante el servicio (Failed) Access 4 acabará su paso por el proceso en Time.Failed.Closed (0,08)	E11
1,52	Access 4 acaba su paso por el proceso	E12
1,69	Llega un customer tipo 1 slightly ill (Access 5) Access 5 pasa a allocated pero el Medical Intern se toma un descanso (break) Access 5 pasará de nuevo a allocated al finalizar el descanso del médico	E13
1,74	Access 5 pasa de nuevo a Allocated Pasará a In Service	E14
1,77	Access 5 pasará a Served en Service Time (0,22)	E15
1,79	Access 3 pasa a Served y terminará su paso por el proceso en Time Served Closed	E16
1,81	Access 3 termina su paso por el proceso	E17
1,96	Access 5 pasa a Served y terminará su paso por el proceso en 0,02	E18
1,98	Access 5 termina su paso por el proceso	E19
2,13	Llega un Customer tipo 1 Slightly ill (Access 6) y pasa a Allocated Pasará a In Service en 0,03	E20
2,16	Access 6 pasa a In Service Access 6 pasará a Served en Service Time	E21
2,29	Access 6 pasa a Served y terminará su paso por el proceso	E22
2,31	Access 6 termina su paso por el proceso	E23
2,55	Llega un Customer tipo 1 Slightly ill (Access 7, ID_Allocation 1) y pasa a Allocated Se le asigna un Cardiologist y pasará a In Service en 0,03	E24
2,58	Access 7 pasa a In Service y el servicio es interrumpido Tras el tiempo de interrupción pasará a Allocated de nuevo	E25
2,64	Access 7 pasa a In Service y pasará a Served tras Service Time ()	E26
2,73	Llega un Customer tipo 1 Slightly ill (Access 8) y accede a un Medical Intern Pasa a Allocated y pasará a In Service en Time.Allocated.InService	E27
2,76	Access 8 pasa a In Service y pasará a Served en Service Time	E28
2,79	Access 7 pasa a Served y terminará su paso por el proceso	E29
2,81	Access 7 termina su paso por el proceso	E30
3,04	Llega un customer tipo 2 Seriously ill (Access 9) y accede a un Cardiologist por ambulancia Pasa a Allocated y pasará a In Service en Time.Allocated.InService	E31
3,05	Llega un customer tipo 1 slightly ill (Access 10) y accede a un Medical Intern Pasa a Allocated pero el médico se toma un descanso Pasará a allocated de nuevo después del descanso	E32
3,09	Access 9 pasa a In Service y pasará a Served después de Service Time	E33
3,1	Access 10 pasa a allocated y pasará a In Service	E34
3,13	Access 10 pasa a In Service y pasará a Served después de Service Time	E35
3,14	Llega un customer tipo 2 Seriously ill (Access 11) Llega por ambulancia y pasa a Waiting list para ser atendido por el cardiologist	E36
3,26	Access 8 pasa a Served y terminará su paso por el proceso	E37

3,28	Access 8 termina su paso por el proceso	E38
3,29	Llega un customer tipo 2 Seriously ill (Access 12) Pasa a Waiting List a la espera de ser atendido por un Cardiologist	E39
3,37	Llega un customer tipo 1 Slightly ill (Access 13) Pasa a Allocated con un Medical Intern y pasará a In Sercive	E40
3,4	Access 9 pasa a Served y terminará su pao por el proceso El Cardiologist que queda libre atiende a Access 11 que pasa a allocated Access 11 pasará a In Service Access 13 pasa a In Service y pasará a Served en Service Time	E41
3,42	Access 13 pasa a Served y terminará su paso por el proceso	E42
3,44	Access 9 termina su paso por el proceso Access 13 termina su paso por el proceso	E43
3,45	Access 11 pasa a In Service y pasará a Served después de Time Service Llega un customer tipo 1 Slightly ill (Access 14) Access 14 pasa a allocated con un Medical Intern y pasará a Served tras Service Time	E44
3,47	Access 10 pasa a served y terminará su paso por el proceso	E45
3,48	Access 14 pasa a In Service y el servicio es interrumpido durante un tiempo Tras la interrupción, Access 14 volverá a In Service	E46
3,49	Access 10 termina su paso por el proceso	E 47
3,54	Access 14 pasa a In Service y pasará a seved tras el servicio	E48
3,59	Llega un customer tipo 1 Slightly ill (Access 15) y pasa a allocated El Medical Intern se toma un descanso y el customer volverá a allocated tras el descanso	E49
3,64	Access 15 pasa a allocated de nuevo y pasará a In Service	E50
3,67	Access 15 pasa a In Service pero el servicio es interrumpido El servicio se reanudará tras el tiempo de interrupción	E51
3,73	Access 15 pasa a In Service y pasará a Served tras el tiempo de servicio	E52
3,75	Llega un customer tipo 2 Seriously ill (Access 16) Llega por Ambulancia para ser atendido por un Cardiologist, pero pasa a sala de espera En la lista de espera muere	E53
3,83	Access 16 termina su paso por el proceso Llega un customer tipo 1 slightly ill (Access 17) Solicita Medical Intern y pasa a Waiting List Al estar en WL, se toma un descanso y se va a la cafetería. Pasará a WL.	E54
3,96	Access 14 pasa a served y termina su servicio. Terminará el paso por el proceso	E55
3,98	Access 14 termina su paso por el proceso	E56
4,02	Access 11 pasa a Served y Terminará su paso por el proceso Al quedar un cardiologist disponoble, Access 12 pasa de WL a Allocated Access 12 pasará a In Service	E57
4,07	Access 11 termina su paso por el proceso Access 12 pasa a In Service y pasará a Served después del tiempo de servicio	E58
4,1	Access 17 termina el descanso en la cafetería y vuelve a WL Access 17 pasa a Allocated pued un Medical Intern está disponible y pasará a In Service	E59
4,13	Access 17 pasa a In Service y pasara a Seved	E60
4,14	Access 15 pasa a Served y terminará su paso por el proceso	E61
4,16	Access 15 acaba su paso por el proceso	E62
4,17	Llega un Customer tipo 1 Slightly ill (Access 18) Pasa a Allocated con un Medical Intern y pasará a In Sercive	E63
4,18	Llega un customer tipo 2 Seriously ill (Access 19) Pasa a lista de espera pues el cardilogist está ocupado	E64
4,2	Access 18 pasa a In Service y el servicio se interrumpe Tras un tiempo de interrupción, volverá a In Service	E65
4,21	Llega un customer tipo 1 Slightly ill (Access 20) y pasa a WL Se toma un descanso en la cafetería por estar en WL Tras el descanso pasará de nuevo a WL	E66
4,23	Access 17 pasa a Served y terminará su paso por el proceso	E67
4,25	Access 17 termina su paso por el proceso	E68
4,26	Access 18 pasa a In Service y pasará a Served	E69
4,38	Access 12 pasa a Served y terminará su paso por el proceso Access 19 que estaba en WL pasa a Allocated con el cardiólogo disponible Access 19 pasará a In Service Llega un customer tipo 1 Slightly ill (Access 21) y pasa a allocated El Medical Intern se toma un descanso, tras el cual Access 21 volverá a allocated	E70
4,4	Llega un customer tipo 2 Seriously ill (Access 22) y pasa a WL El siguiente customer tipo 2 llegaría más allá del límite de tiempo (No entra)	E71
4,41	Access 20 vulve a WL tras su paso por la cafetería	E72
4,42	Access 12 termina su paso por el proceso	E73
4,43	Access 19 pasa a In Service y pasará a Served tras Service Time Access 21 vuelve a Allocated tras el descanso del Medical Intern y pasará a In Service	E74
4,46	Access 21 pasa a In Service y pasará a Served	E75
4,48	Access 21 pasa a Served y terminará su paso por el proceso Access 20 pasa a allocated al quedar libre un Medical Intern, y pasará a In Service	E76
4,5	Access 21 termina su paso por el proceso	E77

4,51	Access 20 pasa a In Service y el servicio es interrumpido Tras la interrupción, Access 20 volverá a In Service	E78
4,57	Access 20 pasa a In Service y pasará a Served tras Service Time	E79
4,6	Access 20 pasa a Served y terminará su paso por el proceso	E80
4,62	Access 20 termina su paso por el proceso	E81
4,76	Access 19 pasa a Served y terminará su paso por el proceso Access 22 pasa a allocated al quedar libre un Cardiologist, y pasará a In Service	E82
4,8	Access 19 termina su paso por el proceso	E83
4,82	Access 22 pasa a In Service y pasará a Served tras Service Time	E84
4,84	Llega un customer tipo 1 Slightly ill (Access 23) y pasa a allocated Access 23 pasará a In service	E85
4,87	Access 18 pasa a Served y terminará su paso por el proceso Access 23 pasa a In Service y el servicio se interrumpe	E86
4,89	Access 18 termina su paso por el proceso	E87
4,93	Access 23 pasa a In Service y pasará a Served	E88
4,94	Llega un customer tipo 1 Slightly ill (Access 24) y pasa a allocated con Medical Intern El siguiente customer tipo 1 llegaría más allá del límite de tiempo (No entra) Access 24 pasará a In Service	E89
4,97	Access 23 pasa a In Service y pasará a Served	E90
4,99	Access 24 pasa a Served y terminará su paso por el proceso	E91
5,01	Access 24 termina su paso por el proceso	E92
5,02	Access 22 pasa a Served y terminará su paso por el proceso	E93
5,06	Access 22 termina su paso por el proceso	E94
5,66	Access 23 pasa a Served y terminará su paso por el proceso	E95
5,68	Access 23 termina su paso por el proceso	E96